VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vizualizace dat s vysokou dostupností

High Availability Data Visualization

2020 Bc. Tereza Kovalová

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student:	**Bc. Tereza Kovalová**
Studijní program:	N2647 Informační a komunikační technologie
Studijní obor:	2612T025 Informatika a výpočetní technika
Téma:	Vizualizace dat s vysokou dostupností High Availability Data Visualization
Jazyk vypracování:	čeština

Zásady pro vypracování:

Cílem diplomové práce je navrhnout a vytvořit systém s vysokou dostupností, který bude umožňovat vizualizaci dat prostřednictvím webového rozhraní.

Práce bude zahrnovat následující body:
1. Proveďte rešerši v oblasti vysoké dostupnosti a vizualizace dat.
2. Prostudujte dostupné nástroje pro vizualizaci dat.
3. Navrhněte a implementujte vlastní nástroj pro vizualizaci dat. Implementace aplikace bude provedena v PHP, HTML, CSS a Javascriptu. Implementujte alespoň dva různé způsoby pro načítání dat pro vizualizaci.
4. Navrhněte a nakonfigurujte řešení pro vysokou dostupnost implementovaného nástroje pro vizualizaci dat. Vysoká dostupnost bude realizována pomocí HAProxy a KeepaliveD.
5. Navrhněte a implementujte aplikace pro Arduino se senzory pro měření teploty a vlhkosti s vysokou dostupností. Pro přenos naměřených dat bude použit protokol MQTT.

Seznam doporučené odborné literatury:

[1] Load Balancing with HAProxy: Open-source technology for better scalability, redundancy and availability in your IT infrastructure, ISBN 978-1519073846
[2] Data Visualization: A Successful Design Process , ISBN 1849693471
[3] MQTT, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Seidl, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě 1. dubna 2020 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 1. dubna 2020 .....

Ráda bych na tomto místě poděkovala panu Ing. Davidu Seidlovi, Ph.D. za pomoc a cenné rady při zpracovávání této práce.

Abstrakt

Tato diplomová práce je zaměřena na oblast vizualizace dat s vysokou dostupností, pro níž byl navržen vizualizační nástroj. Jelikož byl nástroj navržen jako webová aplikace, byla pro zajištění vysoké dostupnosti zvolena HAProxy a Keepalived, a zároveň je webová aplikace nasazena na hlavním a záložním webovém serveru s operačním systémem Debian. Zachycení výpadku dat je zajištěno validací. Dále byla navržena aplikace pro platformu Arduino pro měření teploty a vlhkosti s vysokou dostupností, zajištěnou paralelním měřením několika čidly.

Klíčová slova: vizualizace; MQTT; vysoká dostupnost; diplomová práce; Arduino; čidla

Abstract

This master thesis is focused on the field of high availability data visualization, for which a visualization tool was designed. The tool was designed as a web application, HAProxy and Keepalived were chosen to ensure high availability, and the web application is deployed on the main and backup server running under Debian operation system. Capture of data failure is ensured by validation. Furthermore, an application for measuring temperature and humudity with high availability, ensured by parallel measurement by several sensors, for the Arduino platform was designed.

Keywords: visualization; MQTT; high availability; master thesis; Arduino; sensors

Obsah

Seznam použitých zkratek a symbolů

DVD	–	Digital Versatile Disc
MQTT	–	MQ Telemetry Transport
MTBF	–	Mean Time Between Failures
MTTR	–	Mean Time To Repair
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Secure
URI	–	Uniform Resource Identifier
QoS	–	Quality of Service
MVC	–	Model-View-Controller
GPS	–	Global Positioning System
ER	–	Entity Relationship
JSON	–	JavaScript Object Notation
PDF	–	Portable Document Format
JPG	–	Joint Photographic Experts Group
PHP	–	Hypertext Preprocessor
MySQL	–	My Structured Query Language
SQL	–	Structured Query Language
VRRP	–	Virtual Router Redundancy Protocol
IoT	–	Internet of Things
SMS	–	Short Message Service
GPL	–	General Public License
MIT	–	Massachusetts Institute of Technology
UTF	–	Unicode Transformation Format
IP	–	Internet Protocol
JS	–	JavaScript
CSS	–	Cascading Style Sheets
GSM	–	Global System for Mobile Communication
USB	–	Universal Serial Bus
SSL	–	Secure Sockets Layer
TCP	–	Transmission Control Protocol
IP	–	Internet Protocol
PNG	–	Portable Network Graphics

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Pojem spolehlivost se používá v různých souvislostech a má celou řadu různých interpretací. Podobně jako řada jiných pojmů prošel složitým historickým vývojem.

Vznik pojmu spolehlivost objektu se datuje přibližně do začátku 40. let minulého století, kdy byly vyvíjeny poměrně složité a koncepčně nové zbraňové systémy. Především se jednalo o raketovou techniku, která byla vyvíjena v Německu. U těchto raketových systémů bylo důležité pro zajištění efektivního bojového nasazení této techniky, aby raketa s vysokou pravděpodobností doletěla ke svému cíli a zasáhla ho. Aby byly splněny veškeré požadavky na spolehlivost těchto systémů, museli se technici systematicky a na vědeckých základech zabývat spolehlivostí těchto systémů, jelikož tradiční postupy výroby a vývoje nezaručovaly potřebnou spolehlivost.[1]

S problémem spolehlivosti různých systémů, přístrojů nebo zařízení se ve svém denním životě setkává téměř každý z nás. Uživatel systému nebo zařízení hodnotí jeho spolehlivost podle toho, zda je s ním spokojen. Aby mohl být uživatel zařízení nebo systému spokojen se spolehlivostí, musí se problémem spolehlivosti podrobně zabývat jejich výrobci a konstruktéři.

V dnešní době se můžeme setkat s mnoha zařízeními nebo systémy, které byly navrženy tak, aby vykonávaly funkci, pro kterou byly vyvinuty s velkou spolehlivostí.

U těchto systémů je nezbytně důležité, aby bylo dosaženo vysoké úrovně spolehlivosti, jelikož selhání u takovýchto systémů může vést ke ztrátám na životech či významným finančním ztrátám.

Příkladem takového systému, u kterého je nežádoucí, aby došlo k selhání je robotický operační systém Da Vinci, který se využívá při operacích lidského těla. Tento systém obsahuje ramena ovládající nástroje a kameru. Pohyb ramen simulujících pohyb lidských rukou v těle pacienta se ovládá pomocí joysticků a vizualizace operačního pole je realizována pomocí stereoskopického zobrazovacího kanálu. Selhání ovládání ramen s nástroji by v tomto případě mohlo mít pro pacienta fatální následky.[2]

Dalším zástupcem systémů, u kterých je vyžadována vysoká spolehlivost jsou systémy, které v případě selhání vypínají jiné systémy nebo systémy, které se používají k řízení letu.

U systémů, které se používají k řízení letu nebo v případě selhání vypínají jiné systémy je velice důležitá vysoká dostupnost. Příkladem takového systému může být systém pro řízení letu rakety.

Příkladem selhání systému je havárie rakety, kdy dne 4. června roku 1996 došlo pouhých 37 sekund po startu k explozi rakety Evropské kosmické agentury Ariane 5. U rakety došlo k selhání primárního počítače inerciálních plošin se v důsledku zahlcení daty, jejichž množství bylo proti Ariane 4 neočekávaně velké což bylo způsobeno vyšším počátečním zrychlením rakety. Raketa sice měla záložní počítač, ale ten byl vybaven stejným softwarem, tak u něj došlo k přesně stejnému selhání zahlcení daty jako u primárního počítače.

Dalším příkladem selhání systému je hávárie Boeingu 737 MAX 8 společnosti Lion Air se 189 lidmi na palubě, který se 29. října roku 2018 krátce po startu z indonéské Jakarty zřítil do moře.

Letadlo mělo jediný a vadný senzor úhlu náběhu. Stabilizační systém MCAS navržený Boeingem pak v důsledku vadného senzoru poslal opakovaně letadlo do střemhlavého letu a posádka nedokázala nad letadlem udržet kontrolu[3].

2 Spolehlivost

Tato kapitola se zabývá spolehlivostí jako obecnou vlastností a popisuje pravděpodobnost poruchy u bloku součástek zapojených do série nebo paralelně a dílčích vlastností spolehlivosti jako jsou bezporuchovost, udržovatelnost, skladovatelnost a bezpečnost.

2.1 Definice spolehlivosti

Spolehlivost je obecná vlastnost systému nebo výrobku, který za určitých podmínek a po určitou dobu splňuje danou funkci.

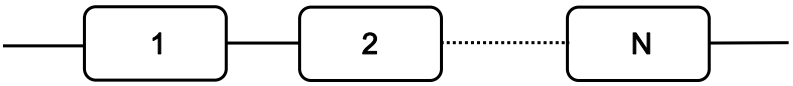
2.2 Teorie spolehlivosti

Teorie spolehlivosti se zabývá matematickými a technickými otázkami spolehlivosti. Hlavními nástroji v teorii spolehlivosti jsou matematická statistika a teorie pravděpodobnosti. Výsledky teorie spolehlivosti lze uplatnit při porovnávání různých variant řešení, k předpovědi chování složitých zařízení a optimalizaci plánů jejich údržby nebo při návrhu samotného zařízení a jeho provozu na požadované úrovni spolehlivosti.

2.3 Bezporuchovost zapojení součástek

2.3.1 Sériové zapojení

Zapojení prvků do série je nejjednodušším typem uspořádání zapojení součástek. K poruše zapojení nebo schopnosti plnit požadované funkce dojde, nastane-li porucha jakékoliv součástky zapojení. Na obrázku 1 je znázorněno sériové zapojení součástek. Sériové zapojení je v bezporu-



Obrázek 1: Sériové zapojení součástek

chovém stavu v případě, že jsou v daném okamžiku v bezporuchovém stavu všechny jeho prvky. Sériové zapojení je v poruchovém stavu v případě, že se nachází v poruchovém stavu alespoň jeden z jeho prvků.

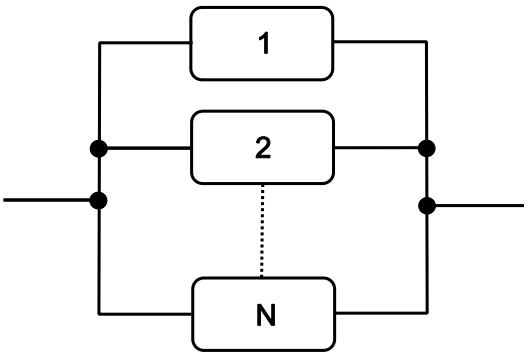
Následuje příklad výpočtu spolehlivosti bloku tvořeného dvěma sériově zapojenými součástkami. Každá součástka v tomto bloku má pravděpodobnost poruchy 20%. Vzorec pro výpočet pravděpodobnosti poruchy bloku součástek pro tento příklad:[5]

$$P(b) = 1 - (1 - P(s))^2 \,[\%] \tag{1}$$

Po dosazení do vzorce uvedeného výše je výsledná pravděpodobnost, že dojde dojde k poruše bloku 36%, i když je pravděpodobnost, že dojde k poruše součástky jen 20%.

2.3.2 Paralelní zapojení

Paralelní zapojení je druhým častým zapojením součástek. K poruše tohoto zapojení dojde v případě, že v jsou v daném okamžiku v poruše všechny jeho součástky. Paralelní zapojení bloku součástek je znázorněno na obrázku 2. Paralelní zapojení se nachází v bezporuchovém



Obrázek 2: Paralelní zapojení součástek

stavu jestliže je v bezporuchovém stavu alespoň jedna jeho součástka. Paralelní zapojení je v poruchovém stavu v případě, že se v poruchovém stavu nacházejí všechny jeho součástky.

Následuje příklad výpočtu spolehlivosti bloku tvořeného dvěma paralelně zapojenými součástkami. Každá součástka v tomto bloku má pravděpodobnost poruchy 20%. Spolehlivost bloku paralelně zapojených součástek se počítá v procentech. Vzorec pro výpočet pravděpodobnosti poruchy bloku součástek pro tento příklad:[5]

$$P(b) = (P(s))^2 \,[\%] \tag{2}$$

Po dosazení do vzorce uvedeného výše vyjde pravděpodobnost, že dojde k poruše bloku 4%, i když je pravděpodobnost, že dojde k poruše součástky 20%.

Z hlediska spolehlivosti je spolehlivější paralelní zapojení součástek, jelikož pro jeho fungování stačí, když je v bezporuchovém stavu jen jedna součástka, kdežto u sériového zapojení musí být funkční všechny součástky. Paralelní zapojení je spolehlivější a zajišťuje vysokou dostupnost.

2.4 Dílčí vlastnosti spolehlivosti

Spolehlivost je charakterizována dílčími vlastnostmi jako je bezporuchovost, udržovatelnost, skladovatelnost, bezpečnost a další.[11]

- **bezporuchovost** je vlastnost výrobku za stanovených podmínek a stanovené době poskytovat požadované funkce bez poruchy
- **udržovatelnost** je vlastnost výrobku, která předepsanou údržbou předchází poruchám[11]
- **skladovatelnost** je vlastnost výrobku, která při dodržení předepsaných podmínek zachovává při skladování a přepravě výrobek v bezporuchovém stavu[11]

- **bezpečnost** je vlastnost výrobku, která po stanovenou dobu a ve stanovených podmínkách neohrožuje životní prostředí nebo lidské životy[11]

3 Porucha

Tato kapitola popisuje definici poruchy, dobu trvání poruchy a rozdělení poruch do různých skupin jako je rozdělení podle vzniku poruchy, časového průběhu poruchy, rozsahu, souvislosti s jinými poruchami nebo podle doby trvání poruchy.

3.1 Definice poruchy

Porucha je změna vlastností nebo jejich částečná nebo úplná ztráta, která způsobuje nemožnost, nebo snižuje schopnost výrobku plnit jeho funkci[11].

3.2 Klasifikace poruch

3.2.1 Podle podmínek vzniku poruchy

Poruchy se dělí podle podmínek vzniku na poruchy z vnitřních a poruchy z vnějších příčin[11].

Poruchy z vnitřních příčin vznikají při dodržení stanovených provozních předpisů a podmínek. Mezi tyto poruchy patří poruchy vznikající v počátku provozu výrobku a označují se jako časné poruchy. Vznik časných poruch je důsledek návrhu a výroby produktu. Další typ poruch vznikajících z vnitřních příčin jsou poruchy dožitím, které vznikají v důsleku opotřebení a stárnutí[11].

3.2.2 Podle časového průběhu

Poruchy se podle časového průběhu dělí na poruchy náhlé, u kterých se změna parametrů výrobku projeví náhlou změnou a na poruchy postupné, u kterých se změna parametrů výrobků projevuje postupně. K postupné poruše může například dojít v důsledku stárnutí nebo opotřebování výrobku[11].

3.2.3 Podle rozsahu

Částečná porucha neboli degradační porucha znamená, že se jedná o poruchu, která nebrání výrobku v plnění požadované funkce. U degradační poruchy se jeden nebo více parametrů odchýlil od úrovně stanovenou technickými podmínkami[11].

Úplná porucha neboli havarijní porucha, je porucha při které výrobek není schopen plnit požadovanou funkci[11].

3.2.4 Podle souvislosti s jinými poruchami

Poruchy podle závislosti se dělí na závislé a nezávislé. Závislá porucha vzniká v důsledku jiné poruchy a nezávislá porucha vzniká nezávisle na jiné poruše[11].

3.2.5 Podle doby trvání

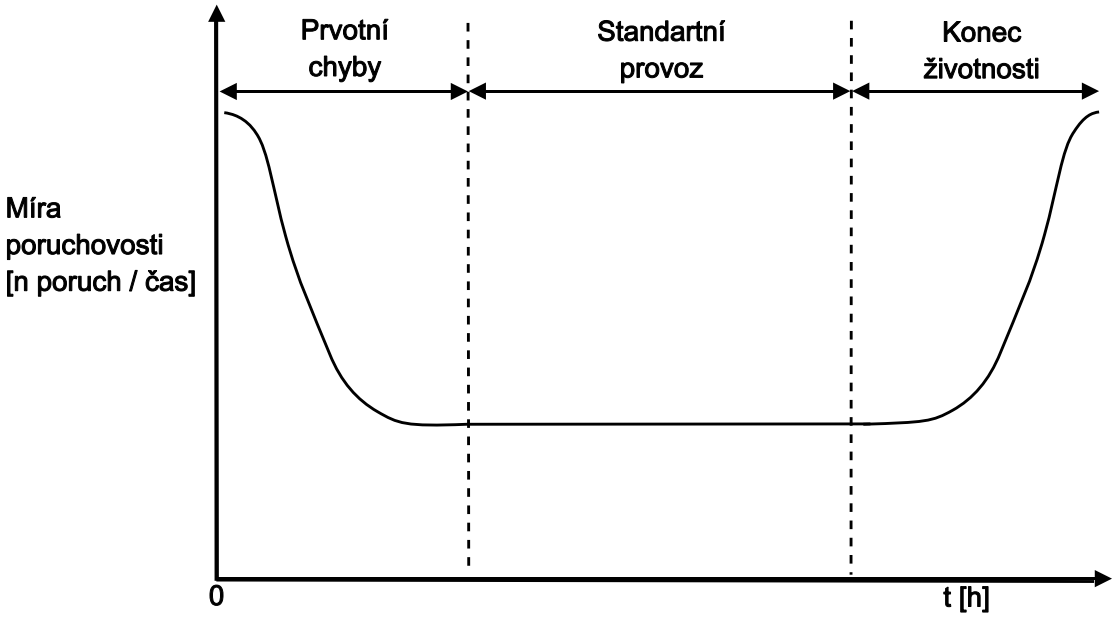
Podle doby trvání se dělí poruchy na dočasné a trvalé. Trvalé poruchy lze odstranit pouze výměnou nebo opravou porouchaného prvku. To znamená, že daný prvek je porouchaný trvale. Dočasné poruchy trvají pouze po dobu působení vnějšího vlivu a může dojít k samovolnému vymizení této poruchy[11].

3.3 Závada

Závadou se označuje stav výrobku, který vede ke zhoršení schopnosti provozu výrobku, ale nezpůsobí ještě poruchu[11].

3.4 Doba do poruchy

Doba do poruchy výrobku je doba od okamžiku, kdy neporouchaný výrobek začal za určitých podmínek pracovat do doby, kdy došlo u výrobku k poruše[11].



Obrázek 3: Křivka četnosti výpadků znázorňující míru poruchovosti[18].

Na obrázku 3 je zobrazena křivka výpadků znázorňující míru poruchovosti. Křivka je rozdělena do tří časových úseků. V prvním časovém úseku jsou zobrazeny prvotní chyby, jejichž četnost je sestupná. Druhý časový úsek znázorňuje standardní provoz, v němž je četnost chyb konstantní. Třetí časový úsek znázorňuje konec životnosti, kde je četnost chyb narůstající.

4 Dostupnost

Tato kapitola popisuje dostupnost, metriky používané pro měření dostupnosti jako je střední doba mezi poruchami a průměrná doba na zotavení. Dále jsou v této kapitole rozděleny a popsány příčiny neplánovaných a plánovaných prostojů. V poslední části této kapitoly jsou popsány dva nástroje, které slouží ke zvýšení vysoké dostupnosti služeb poskytovaných na počítačové síti.

4.1 Popis dostupnosti

Dostupnost lze považovat, za pravděpodobnost, že systém nebo součást systému v daném časovém okamžiku a za určených podmínek umožňuje provádět požadované funkce.

Systémy s vysokou dostupností jsou takové systémy, které jsou nakonfigurovány tak, aby poskytovaly téměř nepřetržitou dostupnost. Systémy s vysokou dostupností obsahují obvykle nadbytečný hardware a software, díky kterému je tento systém dostupný i v případě vyskytnutí poruchy.

V případě, že dojde u systému s vysokou dostupností k poruše, tak proces převzetí služeb při selhání přesune provádění zpracování selhanou komponentou na záložní komponentu. Tento proces obnovuje celosystémové zdroje, neúspěšné nebo dílčí transakce a obnovuje normální stav systému v co nejkratší době, nejlépe v řádu mikrosekund. Čím je u takového systému transparentnější převzetí služeb záložní komponentou v případě selhání pro uživatele, tím je dostupnost systému vyšší.

4.2 Měření vysoké dostupnosti

Pro měření vysoké dostupnosti se obecně používají dva typy metrik:

- střední doba mezi poruchami (MTBF - Mean Time Between Failures)
- průměrná doba na zotavení (MTTR - Mean Time To Repair)

4.2.1 Střední doba mezi poruchami

Střední doba mezi poruchami je základní veličinou pro měření spolehlivosti systému. Většinou se hodnota této veličiny udává v hodinách. Čím vyšší je hodnota střední doby mezi poruchami, tím je daný systém spolehlivější.

Hodnota střední doby mezi poruchami se může odhadnout nebo předpovědět.

Existuje několik metod, které tuto hodnotu počítají na základě návrhu systému. Mezi tyto metody patří například metoda předpovědi podle počtu součástí, která se používá k předpovídání spolehlivosti v počátečních fázích vývoje produktu nebo metoda předpovědi podle analýzy namáhání součástí, která se používá později než předchozí metoda a to v době před předáním hardwaru k výrobě.

V případě, že jsou k dispozici podrobné provozní údaje, tak je lepší místo použití některé metody pro předpověď střední doby mezi poruchami použít jednu z metod pro odhad této veličiny, protože provozní údaje reprezentují skutečně zjištěné poruchy.

$$Spolehlivost = e^{-\left(\frac{čas}{MTBF}\right)}\,[\%] \tag{3}$$

4.2.2 Průměrná doba na zotavení

Průměrná doba na zotavení se stejně jako střední doba mezi poruchami většinou udává v hodinách. Průměrná hodnota na zotavení udává čas, za který dojde k obnovení systému po poruše. Na rozdíl od střední doby mezi poruchami tato hodnota neovlivňuje spolehlivost, ale dostupnost. Odhad může být realizován pomocí metody odhadu na základě podobných položek, kde se odhad provádí na základě historických dat podobné položky. Další metoda, která slouží k odhadu je metoda měření provozních dat. Jedná se pravděpodobně o nejpoužívanější metodu výrobci v programu pro řízení kvality.

$$Dostupnost = \frac{MTBF}{(MTBF + MTTR)}[\%] \tag{4}$$

4.3 Příčíny prostojů

Příčiny prostojů se rozdělují do dvou kategorií.

První kategorie příčin prostojů jsou prostoje neplánované. Mezi neplánované příčiny patří chyby dat, systémové chyby, chyby médií a výpadky. Neplánované výpadky je obtížné předpovídat.

Druhou kategorií jsou plánované prostoje. Mezi plánované prostoje patří například pravidelná údržba nebo aktualizace. Zejména ve velkých podnicích, které pokrývají více časových pásem mohou tyto prostoje narušovat provoz.[17]

4.4 Nástroje pro zajištění vysoké dostupnosti

Následující část kapitoly je zaměřena na popis nástrojů, které se používají ke zvýšení dostupnosti služeb poskytovaných na počítačových sítích, protože je nový nástroj na vizualizaci dat navržen jako webová aplikace.

4.4.1 HAProxy

Jedná se o open-source nástroj, který umožňuje směrovat požadavky podle různých pravidel na dostupné servery. HAProxy umožňuje použití více serverů pro jednu aplikaci a v případě výpadku nebo přetížení posílání požadavků na dostupný nebo méně zatížený server[17].

4.4.2 Keepalived

Keepalived je open-source software napsaný v jazyce C, který poskytuje jednoduchý a spolehlivý nástroj pro vyvažování zátěže a vysokou dostupnost pro systémy Linux a infrastrukturu založenou na Linuxu. Vysoká dostupnost je dosažena protokolem VRRP (Virtual Router Redundancy Protocol)[12].

5 Nástroje na vizualizaci

V této kapitole jsou popsány celkem tři nástroje na vizualizaci dat, které jsou vhodné pro vizualizaci v rámci IoT (Internet of Things).

5.1 Grafana

Grafana je analytický a vizualizační multiplatformní open-source nástroj, který umožňuje vizualizování, vyhledávání, upozornění a průzkum uživatelských metrik bez ohledu na to, kde jsou uložená data. Grafana je vhodný nástroj pro vizualizaci dat v čase.

Grafana umožňuje odesílání upozornění prostřednictvím SMS (Short Message Service), e-mailu, Slack a dalších. Do grafů je možné zanést data z více různých zdrojů[20]. Ukázka vizualizace je na obrázku 4.

Výhody:

- široká škála zdrojů
- vlastní anotace
- upozornění a oznámení zasílané emailem
- multiplatformní

5.2 Power BI

Power BI je placený nástroj společnosti Microsoft. Jedná se výkonný nástroj na vizualizaci dat v reálném čase. Umožňuje vizualizovat data z různých zdrojů jako jsou Excel, Google Analytics a další. K nástroji je možné připojit jakékoliv zařízení, senzor nebo aplikaci IoT.[22] Ukázka vizualizace je na obrázku 5.

Výhody:

- streamovaná a statická data
- dotazy na data v přirozeném jazyce

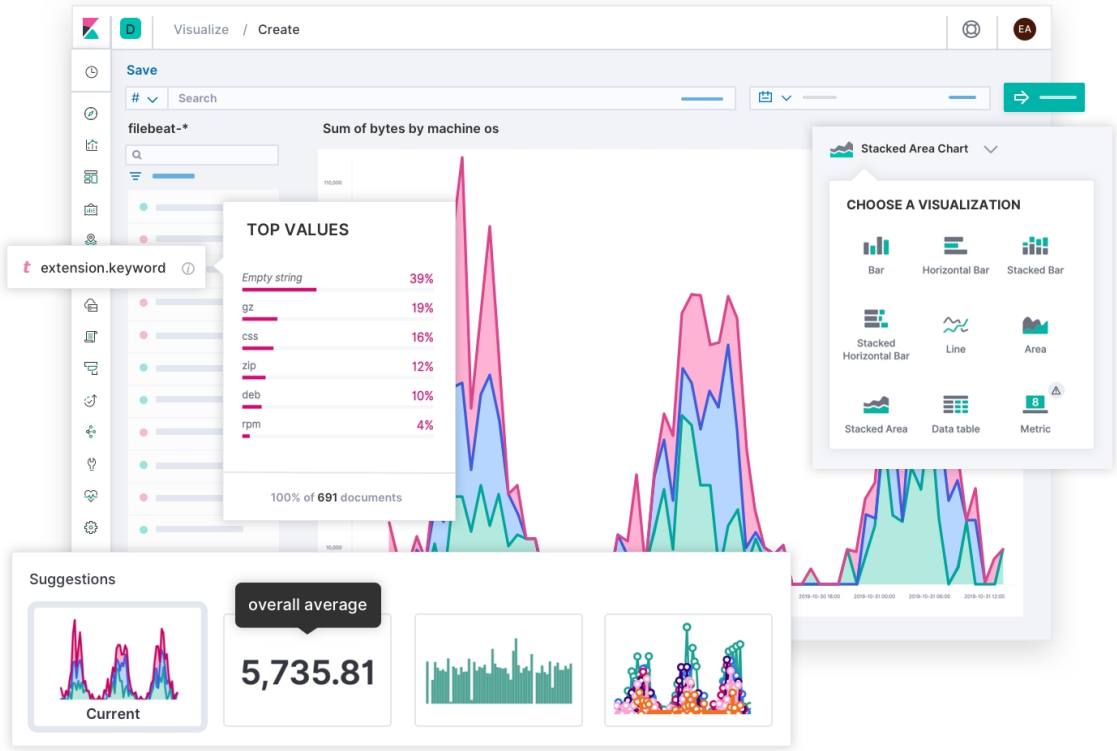
Obrázek 4: Ukázka nástroje Grafana

Obrázek 5: Ukázka nástroje Power BI

5.3 Kibana

Kibana je součástí sady nástrojů pro správu dat Elastic Stack. Kibana kromě pokročilé vizualizace poskytuje funkce pro správu dat včetně techniky strojového učení k detekci a prozkoumávání anomálií v datových sadách[21]. Ukázka vizualizace je na obrázku6.
Výhody:

- open-source
- strojové učení



Obrázek 6: Ukázka nástroje Kibana

6 Technologie použité pro vývoj vlastního řešení

V této kapitole jsou popsány technologie, které byly použity pro vývoj nástroje na vizualizaci dat s vysokou dostupností. První část této kapitoly se věnuje databázi. V druhé části této kapitoly je popsán programovací jazyk, který byl použit pro vývoj nástroje.

6.1 Databáze

Důležitou součástí nástroje pro vizualizaci dat je možnost uložení dat do databáze, aby mohly být následně zpracovány a vizualizovány.

Vzhledem ke zkušenostem s databázemi byla zvolena databáze MariaDB. Jedná se o open-source relační databázi vyvíjenou původními vývojáři MySQL jako vylepšenou náhradu za MySQL (My Structured Query Language). Databáze MariaDB byla použita ve verzi 10.1.35, která spadá pod licenci GPL (General Public License) verze 2.

6.2 Programovací jazyk

Pro vývoj nástroje pro vizualizaci dat s vysokou dostupností byl zvolen skriptovací jazyk PHP (Hypertext Preprocessor) ve verzi 7.2. Hlavním důvodem proč byl zvolen právě tento jazyk byly praktické zkušenosti s vývojem aplikací v tomto skriptovacím jazyce.

Vzhledem k tomu, že v dnešní době existuje pro každý programovací jazyk mnoho frameworků, které zjednodušují a urychlují vývoj aplikací, proto byl pro realizaci nástroje vybrán PHP framework Laravel, který se používá pro vývoj webových aplikací. Laravel byl zvolen proto, že se jedná o open-source framework poskytovaný pod licencí MIT (Massachusetts Institute of Technology), který obsahuje funkcionalitu pro routování, autentizaci uživatelů a mnoho dalších funkcí.

7 Návrh systému

Tato kapitola popisuje návrh nového nástroje pro vizualizaci dat s vysokou dostupností. Je v ní popsána architektura navrhovaného systému a postup zajištění vysoké dostupnosti dat. Hlavním důvodem návrhu a realizace vlastního nástroje pro vizualizaci dat je chybějící vizualizace pro vizualizaci s vysokou dostupností v případech, kdy dojde k výpadku čidla.

7.1 Architektura aplikace

Data mohou být do systému předávána pomocí HTTP (Hypertext Transfer Protocol) protokolu a metody GET nebo pomocí MQTT (MQ Telemetry Transport) protokolu prostřednictvím MQTT klienta. Zdrojem dat pro systém jsou čidla postavená na platformě Arduino, která přenášejí data ve formátu JSON (JavaScript Object Notation) a MQTT prostřednictvím protokolu. Jako zdroj dat předávaných pomocí MQTT protokolu by bylo možné použít i jinou platformu než je Arduino za předpokladu, že by byla dodržena struktura formátu JSON.

Systém přijatá data ukládána do databáze. Data jsou později validovaná a záznam s daty v databázi je doplněn o stav validace. Validace reaguje na detekované chyby odesláním notifikací. Pro rychlou identifikaci případných problémů lze stav validace vizualizovat.

Systém je navržený tak, aby bylo možné měřit jednu veličinu ve stejný čas více čidly. Takto navržená architektura umožňuje eliminaci rizika výpadku čidla. Architektura aplikace je zobrazena na obrázku 7.

Nástroj pro vizualizaci dat je navržen jako webová aplikace s MVC (Model-View-Controller) architekturou.

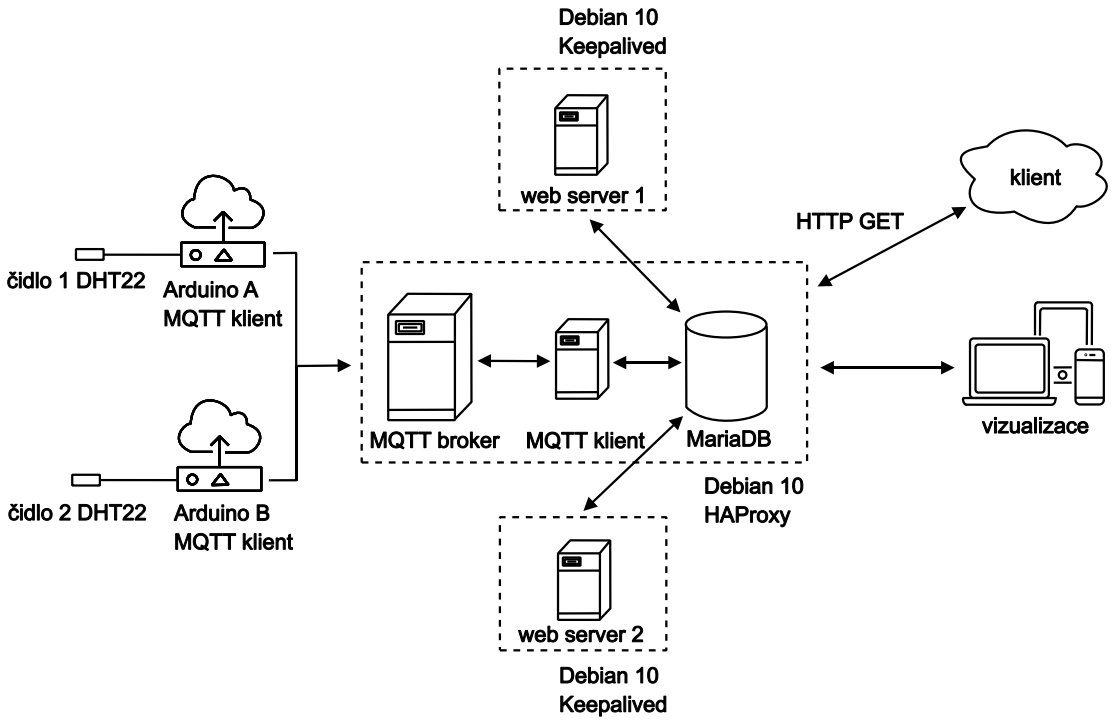
Architektonický vzor MVC je velice oblíbený pro architekturu aplikací, u kterých existuje více způsobů pro interakci s daty a jejich zobrazení. Tento vzor rozděluje aplikaci do třech logických komponent:

- **model** - spravuje systémová data a přidružené operace s těmito daty
- **view** - definuje jakým způsobem se data prezentují uživateli
- **controller** - obsluhuje interakce uživatele a předává je view a modelu

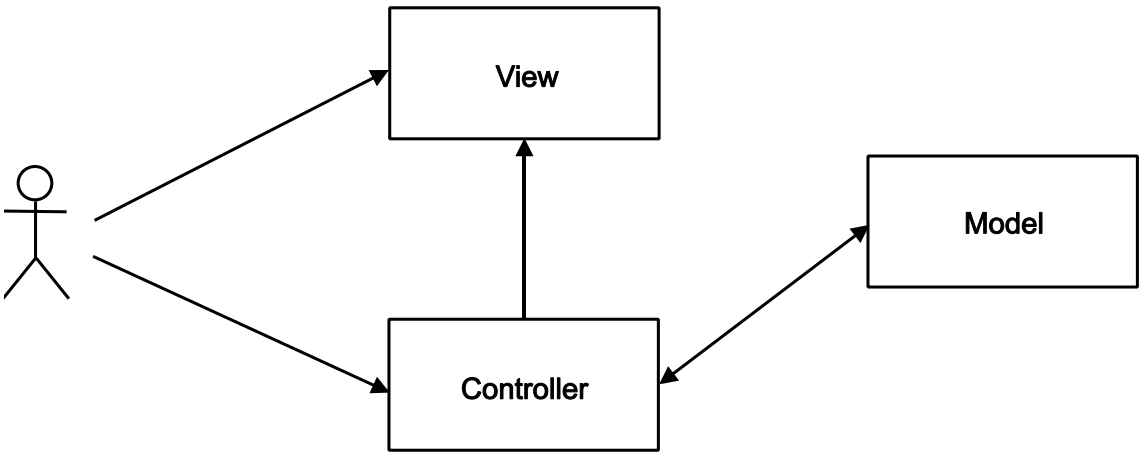
Tyto tři logické komponenty umožňují oddělení prezentace a interakce od systémových dat. Výhodou této architektury je, že umožňuje změnu dat nezávisle na jejich reprezentaci a naopak[4].

7.2 Vysoká dostupnost dat

Zajištění vysoké dostupnosti dat je možné pomocí paralelního měření hodnot ve stejný čas. Vysoká dotupnost dat je pak zajištěna pomocí validace, která je popsána v kapitole 9.1.7.



Obrázek 7: Architektura aplikace



Obrázek 8: MVC

7.3 Přenos dat

Pro přenos dat pro vizualizaci jsou zvoleny dva protokoly. První protokol je protokol MQTT, který se hojně využívá v IoT pro přenos dat mezi čidly a nadřazenými systémy. Protokol MQTT je popsán v kapitole 7.3.2. Druhý zvolený protokol pro přenos dat do aplikace je protokol HTTP a je popsán v kapitole 7.3.1.

7.3.1 Hypertext Transfer Protocol

HTTP je aplikační objektově orientovaný protokol, který pracuje na principu požadavek a odpověď znázorněný na obrázku 9. Protokol HTTP není bezpečný a proto se používá společně SSL (Secure Sockets Layer), jako HTTPS (Hypertext Transfer Protocol Secure). Klient naváže spojení se serverem a vyšle požadavek. Tento požadavek může obsahovat informace o klientovi a data jako je verze protokolu[16].

HTTP protokol obsahuje několik funkcí:

- GET - používá se k načtení dat ze serveru
- POST - používá se k odesílání dat na server
- PUT - používá se k nahrazení všech aktuálních reprezentací cílového zdroje nahraným obsahem
- HEAD - stejná jako GET, ale přenáší pouze stavový řádek a část záhlaví
- DELETE - slouží k odstranění cílového zdroje

Ze všech funkcí, které obsahuje HTTP protokol byla pro přenos dat z libovolného zdroje do nástroje zvolena metoda GET, a pro přenos notifikací z nástroje do libovolného cíle metoda POST.

```
https://www.fantasyobchod.cz/index.php?category_id=123&sort=price-asc
```

Výpis 1: Příklad předávání proměnných u požadavku GET

Ve výpisu 1 je uveden příklad předávání proměnných u požadavku GET. Od jednotného identifikátoru zdroje (URI - Uniform Resource Identifier) jsou proměnné oddělené znakem ? a jednotivé proměnné jsou mezi sebou odděleny znakem &. V příkladu jsou dvě proměnné. První proměnná category_id má hodnotu 123 a druhá proměnná má hodnotu price-asc.

Zprávy HTTP protokolu obsahují pole hlavičky, které se mohou používat v požadavcích a odpovědích. Některé z těchto polí jsou výhradně určeny pro požadavky klienta a některé zase výhradně pro odpovědi serveru. Následuje seznam a popis některých polí hlaviček.

- **Allow** - identifikuje, které HTTP metody server podporuje

- **Authorization** - pole využívají servery, které nedovolují anonymní přístup k některým informacím, v poli jsou autentikační informace uživatele
- **Content-Encoding** - pole slouží k identifikaci typu použitého kódování přenášeného obsahu
- **Content-Length** - pole specifikuje velikost těla zprávy
- **Content-Type** - pole specifikuje MIME typ média těla zprávy
- **Date** - pole definuje datum a čas vzniku zprávy
- **Expires** - pole obsahuje datum a čas, po kterém jsou data neplatná
- **Server** - pole je součástí odpovědi serveru a specifikuje, který HTTP server odpovídá na požadavky
- **User-Agent** - pole se odesílá jako součást požadavku a obsahuje informace o typu webového prohlížeče
- **Accept-Charset** - pole definuje, kterou znakovou sadu má server použít v odpovědi
- **Accept-Encoding** - pole definuje, jaké kódování má server použít v odpovědi
- **Accept-Language** - pole určuje, které jazyky jsou preferované v odpovědi
- **Accept-Ranges** - pole umožňuje serveru indikovat rozsah přijímaných dotazů
- **Age** - pole sděluje odesilateli odhad času potřebného k vygenerování odpovědi na serveru
- **Connection** - pole umožňuje odesilateli specifikovat nastavení pro vlastní připojení
- **Host** - pole specifikuje hosta, který odesílá požadavek
- **Public** - pole obsahuje seznam metod podporovaných serverem
- **Warning** - pole se používá k přenosu dodatečných informací o stavu odpovědi, které nemohou být odvozeny od stavového kódu

```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

Výpis 2: Příklad požadavku GET

Ve výpisu 2 je uveden příklad požadavku GET. Na prvním řádku je uvedeno, že se jedná o HTTP metodu GET, identifikátor zdroje je /hello.htm a byl použit HTTP protokol verze 1.1. Na druhém řádku je uvedena informace o typu použitého uživatelského agenta (webového prohlížeče). Na třetím řádku je specifikován zdroj, který odesílá požadavek. Na čtvrtém řádku je specifikovaný jazyk, který je preforovaný v odpovědi. Na pátém řádku jsou uvedeny typy kódování, které jsou požadovány v odpovědi. Na posledním řádku je specifikováno nastavení pro vlastní připojení.

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

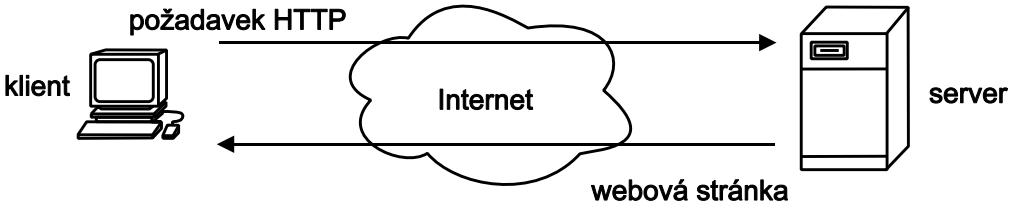
Výpis 3: Odpověď serveru na požadavek GET

Ve výpisu 3 je uveden příklad odpovědi na GET požadavek. Na prvním řádku výpisu je uvedeno, že se jedná o HTTP protokol verze 1.1 a požadavek byl vyřízen se stavem 200, s doplňující informací OK. Na druhém řádku je uvedeno datum a čas, kdy zpráva vznikla. Na třetím řádku je uvedeno jaký HTTP server odpovídá na požadavky. Na čtvrtém řádku je uvedeno datum a čas, kdy byl požadovaný dokument naposledy upraven. Na pátem řádku je uveden rozsah přijímaných dotazů. Na šestém řádku je uvedena velikost těla zprávy. Na sedmém řádku je uveden typ média těla přenášené zprávy. Na posledním řádku je specifikováno nastavení pro vlastní připojení.

V hlavičce odpovědi serveru na požadavek klienta se posílá stavový kód. Tento kód udává s jakým výsledkem byl požadavek vyřízen. Stavové kódy se rozdělují do celkem pěti skupin:

- **1xx** - kódy začínající číslicí 1 jsou informační kódy
- **2xx** - kódy začínající číslicí 2 jsou kódy pro úspěšné vyřízení požadavků
- **3xx** - kódy začínající číslicí 3 jsou kódy pro přesměrování
- **4xx** - kódy začínající číslicí 4 jsou kódy pro chybu na straně klienta
- **5xx** - kódy začínající číslicí 5 jsou kódy pro chybu na straně serveru

Na obrázku 9 je znázorněna komunikace mezi klientem a HTTP serverem. Klient vytvoří požadavek například tím, že uživatel zadá internetovou adresu do webového prohlížeče a prohlížeč odešle požadavek na webový server. Server požadavek zpracuje a odešle požadovaná data zpět prohlížeči, který je zobrazí uživateli.



Obrázek 9: Komunikace po HTTP

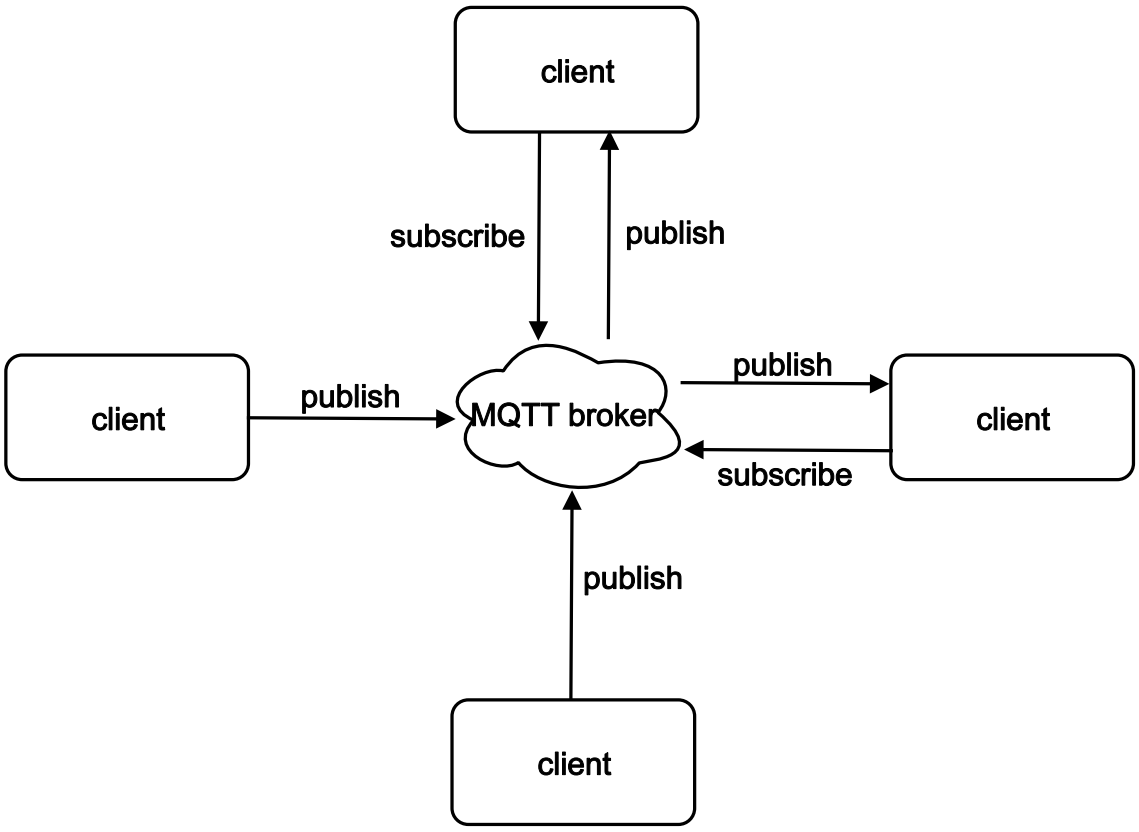
7.3.2 MQTT (Message Queuing Telemetry Transport)

MQTT je jednoduchý a nenáročný protokol postavený nad TCP/IP pro posílání zpráv mezi jednotlivými zařízeními. MQTT definovali v roce 1999 Dr. Andy Stanford-Clark a Arlen Nipper.

Protokol je založený na principu publikování (publish) zpráv a na přihlášení k jejich odběru (subscribe). Zprávy jsou rozdělené do témat (topic) a o jejich výměnu od klienta, který zprávu publikuje ke klientům, kteří jsou přihlášení k odběru zpráv z daného tématu se stará jeden centrální bod (MQTT broker)[9].

MQTT protokol byl zvolen pro přenos naměřených hodnot z čidel do nástroje a pro přenos konfigurace z nástoje do čidel, protože se jedná o jednoduchý a snadno použitelný protokol.

Nasledující obrázek 10 znázorňuje princip předávání zpráv mezi jednotlivými zařízeními a MQTT brokerem.



Obrázek 10: MQTT

Jednotlivé zprávy jsou rozděleny do hierarchické struktury jednotlivých témat. Každá zpráva může patřit pouze jednomu tématu a hierarchie témat se odděluje lomítky. Ve specifikci témat může být použita diakritika, jelikož MQTT používá UTF-8 (Unicode Transformation Format). Zprávy nemají definované, jak by měl vypadat jejich obsah, ale jsou pouze omezeny velikostí, které je definována verzí MQTT protokolu. Zprávy mohou obsahovat například text, hodnotu nebo binární data jako je JSON. Ke každé zprávě jsou přidávány servisní data, která například definují způsob, jakým má být zpráva po doručení potvrzena[8].

Publisher, ten kdo vytváří zprávu, zvolí téma a odešle ho spolu s vytvořenou zprávou. Téma, které publisher zvolí, nemusí nijak zakládat nebo kontrolovat. Subscriber je ten, kdo bude odebírat zprávy a k odběru zpráv se přihlásí brokeru speciální zprávou, která obsahuje subscribe a název tématu, ze kterého chce zprávy odebírat. Publisher může v názvu tématu použít zástupný znak # nebo +. První zástupný znak nahrazuje v hierarchii tématu jednu nebo více úrovní a uvádí se vždy na konci. Druhý zástupný znak nahrazuje v hierarchii tématu jen jednu úroveň. Broker přijme zprávu pro topic, který zvolil publisher a v případě, že takový ještě nemá, tak ho založí. Je důležité, aby si publisher a subcriber předem dohodli, které téma budou používat. Pro přenos zpráv, které potřebuje poslat sám broker jsou speciální témata, která začínají znakem $.

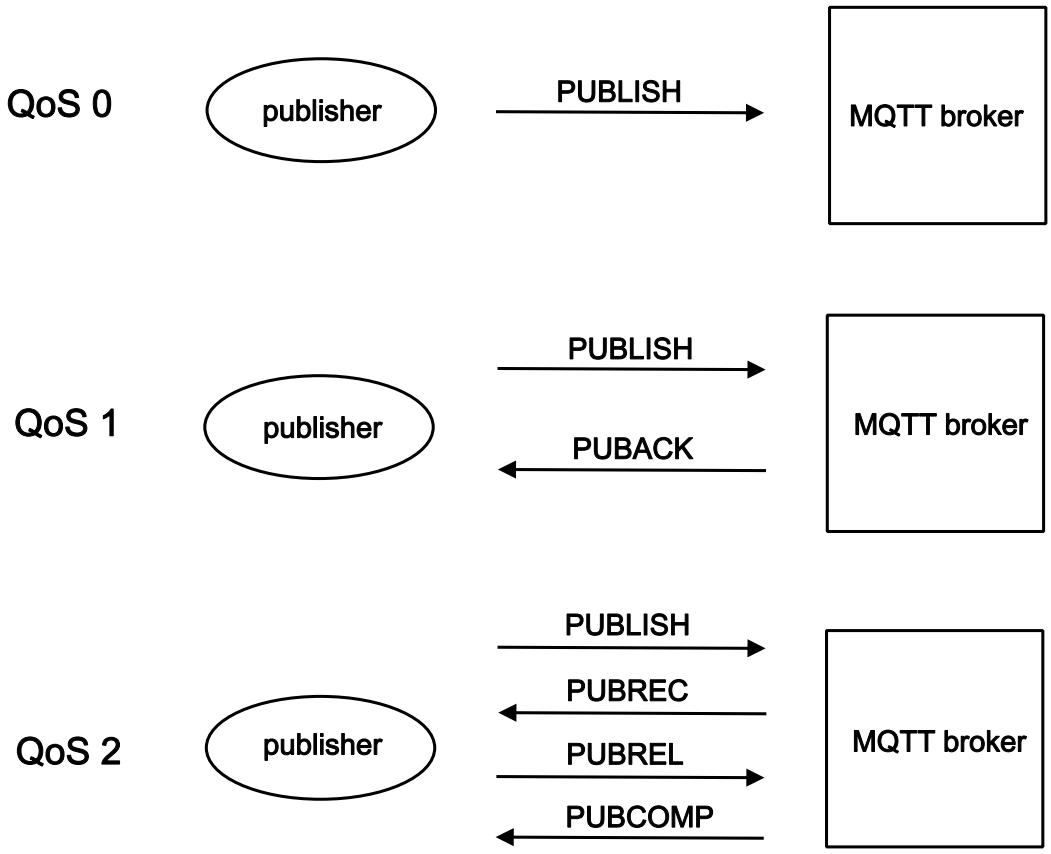
MQTT má tři úrovně QoS (Quality of Service), které určují kvalitu s jakou jsou zprávy doručovány. Tyto tři úrovně jsou znázorněny na obrázku 11. Tyto úrovně se rozlišují podle toho, jak je zajištěno předávání zpráv a do jaké míry jsou tyto zprávy potvrzovány.

Popis jednotlivých úrovní QoS:

- Na úrovni 0 publisher odešle zprávu PUBLISH brokeru a ten ji stejným způsobem odešle všem subscriberům
- Na úrovni 1 publisher odešle zprávu PUBLISH brokeru a čeká na její potvrzení. Broker odešle zprávu všem subscriberům a čeká na jejich potvrzení přijetí zprávy. Broker může potvrdit doručení zprávy, jakmile obdrží potvrzení o přijetí zprávy PUBACK od jednoho nebo všech odběratelů.
- Na úrovni 2 publisher odešle zprávu PUBLISH brokeru. Broker pošle zprávu subscriberům a publisherovi potvrdí přijaté zprávy zprávou PUBREC. Publisher odpoví na zprávu PUBREC zprávou PUBREL. Broker zprávu smaže a potvrdí zprávou PUBCOMP.

Broker odesílá zprávu subscriberu na stejné úrovni jako ji odeslal publisher a v případě, že takovou úroveň subscriber nepodporuje, tak broker zprávu odesílá s nižší úrovní.

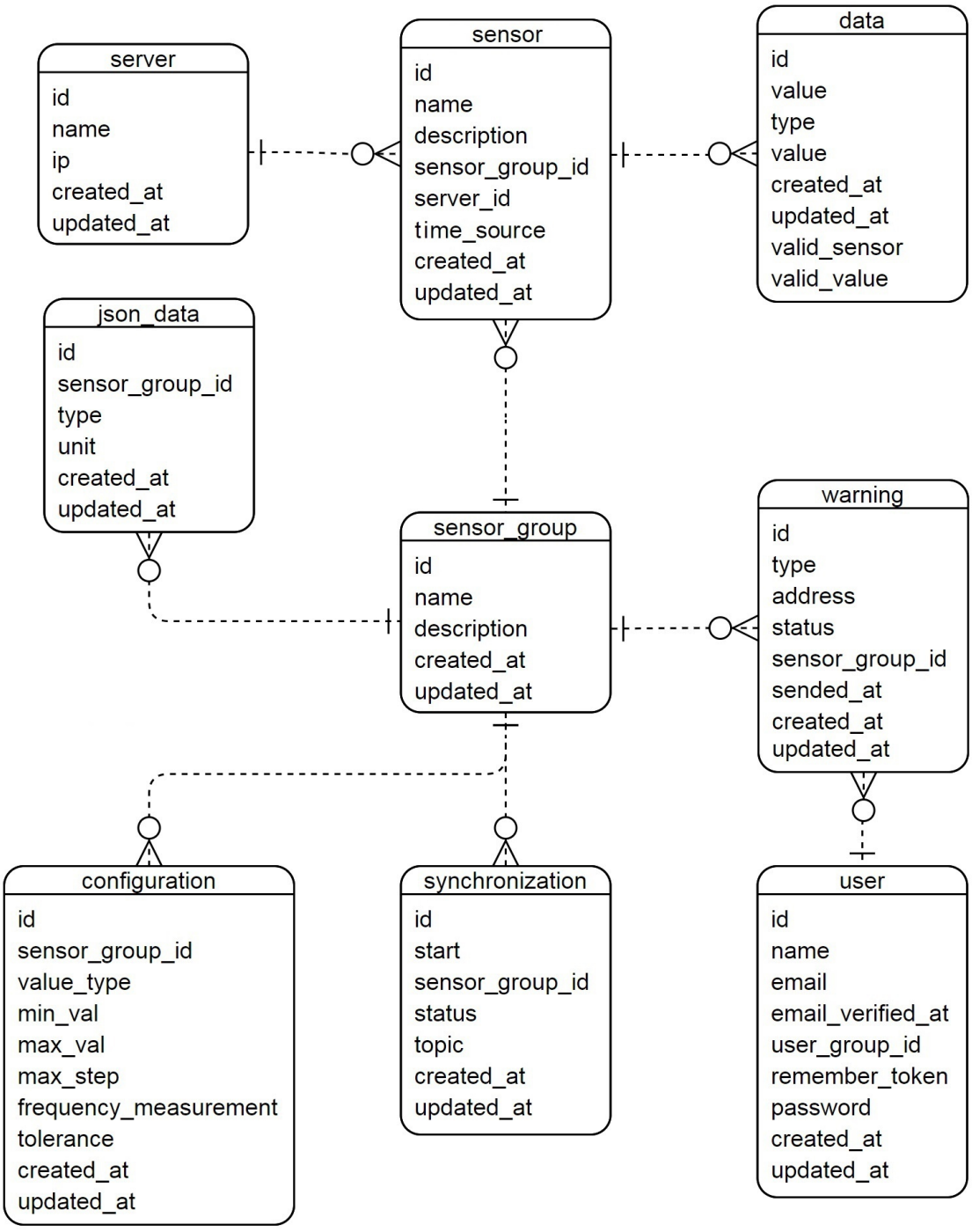
Kromě QoS se také u zpráv nastavuje příznak *retain flag*, který říká brokeru, že po rozeslání zprávy subscriberům nemá zprávu zahazovat, ale uložit a poslat novým subscriberům. Novým subscriberům se odesílá vždy poslední uložená zpráva s tímto příznakem.[10]



Obrázek 11: MQTT QoS

8 Návrh datového modelu

Tato kapitola popisuje entity a atributy jednotlivých entit. Vazby mezi jednotlivými entitami jsou znázorněny na ER (Entity Relationship) diagramu na obrázku 12.



Obrázek 12: ER diagram

8.1 Popis entit

8.1.1 Čidla

Entita sensors reprezentuje čidla, která měří hodnoty, například teplotu a vhlkost, a naměřené hodnoty odesílají pomocí MQTT do MQTT brokeru. Čidlo je přiřazeno k jednomu MQTT brokeru a jedné skupině čidel.

Atribut	Typ
id	int
name	varchar
description	varchar
server_id	int
sensor_group_id	int
created_at	datetime
updated_at	datetime

Tabulka 1: Popis entity Sensor

8.1.2 Servery

Entita servers reprezentuje MQTT brokery, které se přiřazují k čidlům. Primárním klíčem této entity je id. Entita obsahuje atributy, které reprezentují název a IP adresu serveru.

Atribut	Typ
id	int
name	varchar
ip	varchar
created_at	datetime
updated_at	datetime

Tabulka 2: Popis entity Servery

8.1.3 Skupiny čidel

Entita sensor_groups reprezentuje skupiny čidel. Ve skupině jsou čidla, která provádějí totožné měření a umožňují validaci čidel v rámci této skupiny. Primárním klíčem této entity je id. Entita obsahuje atributy, které reprezentují název a popis skupiny čidel.

8.1.4 Konfigurace

Entita configurations reprezentuje konfiguraci pro skupinu čidel. Primárním klíčem této entity je id. U každé skupiny čidel se v konfiguraci ukládají hodnoty pro minimální hodnotu, maximální hodnotu a maximální gradient, které se používají při validaci hodnot a čidel. Typ může být

Atribut	Typ
id	int
name	varchar
description	varchar
created_at	datetime
updated_at	datetime

Tabulka 3: Popis entity Skupiny čidel

například teplota nebo vlhkost. Entita configurations je ve zvtahu s entitou sensor_groups. Kardinalita tohoto vztahu je 1:N, tedy skupina čidel může mít N konfigurací. Z tohoto vztahu vyplývá, že entita obsahuje cizí klíč, který reprezentuje identifikátor skupiny čidel, ke kterým tato konfigurace náleží.

Atribut	Typ
id	int
sensor_group_id	int
value_type	varchar
min_val	float
max_val	float
max_step	float
frequency_measurement	int
tolerance	float
created_at	datetime
updated_at	datetime

Tabulka 4: Popis entity Konfigurace

8.1.5 Data

Entita data reprezentuje jednotlivá data, které naměřila čidla a informaci o stavu validace hodoty a čidla. Primárním klíčem této entity je id. Entita data je ve vztahu s entitou čidla. Kardinalita tohoto vztahu je 1:N, což znamená, že jedno čidlo může mít N dat. Z tohoto vztahu vyplývá, že entita data obsahuje cizí klíč, který reprezentuje čidlo, ke kterému náleží.

8.1.6 JSON Data

Entita JSON data reprezentuje data, která se přenášejí v JSON mezi čidlem a MQTT brokerem. Data obsahují informace o měřené veličině. Entita je ve vztahu s entitou, která rerezentuje skupinu čidel. Kardinalita tohoto vztahu je 1:N. Skupina čidel může mít N dat. Primárním klíčem této entity je id a cizím klíčem je identifikátor skupiny čidel.

Atribut	Typ
id	int
value	float
type	varchar
sensor_id	int
sort_order	int
timestamp	datetime
valid_value	int
valid_sensor	int
created_at	datetime
updated_at	datetime

Tabulka 5: Popis entity Data

Atribut	Typ
id	int
sensor_group_id	int
type	varchar
unit	varchar
created_at	datetime
updated_at	datetime

Tabulka 6: Popis entity JSON Data

8.1.7 Notifikace

Entita warnings reprezentuje jednotlivá upozornění, která se vytvářejí v rámci validace hodnot a čidel. Primárním klíčem této entity je id. Entita je ve vztahu s entitou reprezentující skupiny čidel. Kardinalita tohoto vztahu je 1:N, kde skupina čidel může mít N upozornění. Cizím klíčem je identifikátor skupiny čidel.

Atribut	Typ
id	int
type	varchar
description	varchar
address	varchar
status	int
sensor_group_id	int
sended_at	datetime
created_at	datetime
updated_at	datetime

Tabulka 7: Popis entity Notifikace

8.1.8 Synchronizace

Entita synchronizations reprezentuje synchronizaci začátku měření skupiny čidel. Primárním klíčem této entity je id. Entita je ve vztahu s entitou reprezentující skupinu čidel. Kardinalita tohoto vztahu je 1:N. Skupina čidel může mít N synchronizací. Cizím klíčem této entity je identifikátor skupiny čidel.

Atribut	Typ
id	int
start	datetime
sensor_group_id	int
status	int
topic	varchar
created_at	datetime
updated_at	datetime

Tabulka 8: Popis entity Synchronizace

8.1.9 Uživatel

Entita users reprezentuje registrované uživatele. Primárním klíčem této entity je id. Entita je ve vztahu s entitou, která reprezentuje upozornění a kardinalita tohoto vztahu je 1:N. Uživatel tak může mít N upozornění.

Atribut	Typ
id	int
name	varchar
email	varchar
email_verified_at	datetime
user_group_id	int
remember_token	varchar
password	varchar
created_at	datetime
updated_at	datetime

Tabulka 9: Popis entity Uživatel

8.1.10 Email

Entita email slouží k uložení notifikací, které se odesílají emailem. Entita obsahuje adresu, na kterou se bude email odesílat, předmět, tělo zprávy, datum vytvoření, poslední editace a datum a čas odeslání zprávy.

Atribut	Typ
id	int
email	varchar
subject	varchar
body	varchar
password	varchar
created_at	datetime
updated_at	datetime
sended_at	datetime

Tabulka 10: Popis entity Email

9 Nástroj pro vizualizaci dat s vysokou dostupností

Tato kapitola popisuje aplikaci na vizualizaci dat s vysokou dostupností, popisuje strukturu aplikace a všechny její moduly jako je validace hodnot, validace čidel, vizualizace, notifikace a další.

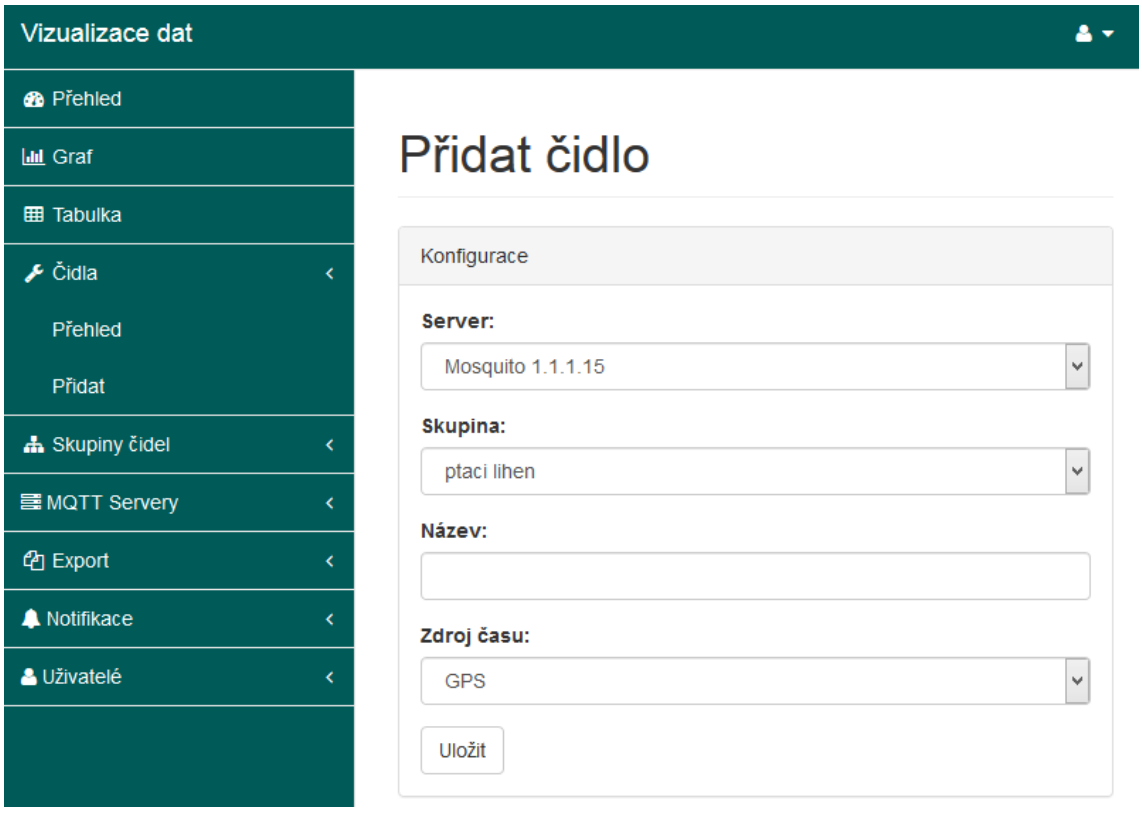
9.1 Popis funkcí

Aplikace pro vizualizaci dat obsahuje několik funkcí. Dostupnost těchto funkcí pro konkrétní skupinu uživatelů je popsána v kapitole 9.4.

9.1.1 Čidlo

Hlavním zdrojem dat pro vizualizaci jsou hodnoty naměřené pomocí čidel. V tomto případě je jako čidlo označeno Arduino se všemi potřebnými moduly pro měření hodnot a jejich odesílání.

Správa čidel v aplikaci obsahuje několik základních funkcí, mezi které patří vložení nového čidla. U čidla se eviduje několik důležitých informací jako je jeho název, popis, server na který čidlo odesílá naměřené hodnoty a informace o tom, ke které skupině čidel patří. Ukázka formuláře pro vložení nového čidla je na obrázku 13.



Obrázek 13: Formulář pro vložení nového čidla

9.1.2 Skupina čidel

Vysoká dostupnost měření je zajištěna paralelním měřením hodnot více čidly najednou, které tvoří skupinu. V rámci skupiny čidel probíhá synchronizace začátku měření.

U skupiny čidel se eviduje jejich název a popis. Správa skupin obsahuje několik základních funkcí pro vložení nové skupiny, editaci skupiny, smazání skupiny a výpis všech vytvořených skupin.

Synchronizace čidel se provádí v rámci celé skupiny a to tak, že se vybere skupina čidel, která se má synchronizovat a zvolí se datum a čas začátku měření. Výběr času začátku měření je možný pouze po celých pěti minutách. Komunikace s čidly probíhá přes MQTT a všechny informace, které se mezi čidlem a systémem přenášejí jsou ve formátu JSON.

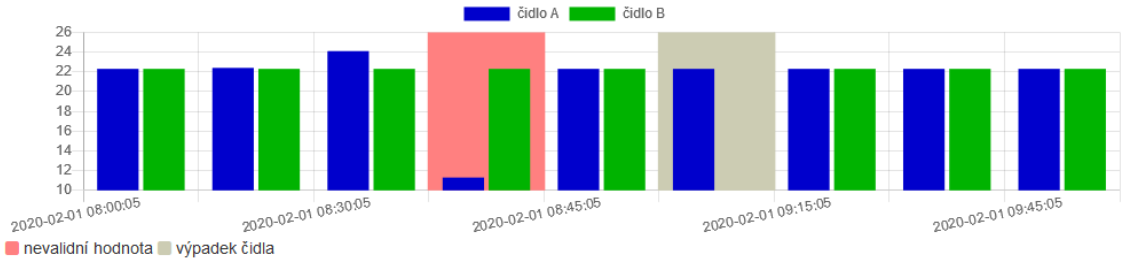
Každá skupina čidel může měřit jinou veličinu a v rozdílném prostředí, pro které je nutné, aby se při validaci hodnot použily různé hodnoty limitů, tak se u každé skupiny nastavuje konfigurace.

9.1.3 Vizualizace

Systém umožňuje vizualizaci dat dvěma způsoby. Data lze vizualizovat pomocí tabulky nebo grafu. Pro oba způsoby vizualizace se používá výběr dat na základě zvoleného časového období, kdy byly hodnoty naměřeny, skupina čidel a typ dat.

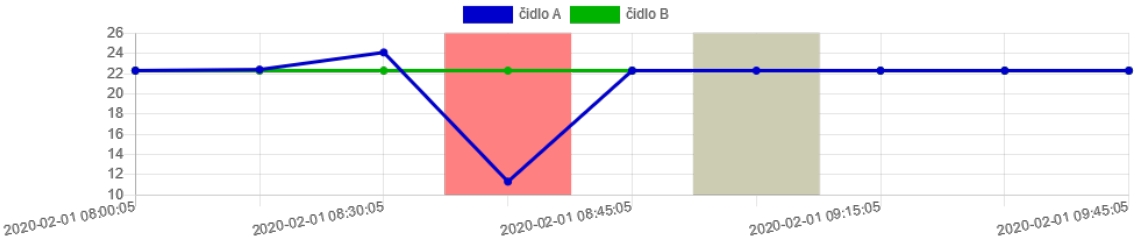
Hodnoty zobrazené v tabulce jsou ze všech čidel zvolené skupiny čidel. V tabulce jsou kromě informace o čidle a naměřené hodnotě také informace o validaci hodnoty a čidla. Hodnoty, které neprošly validací hodnot nebo validací čidla jsou barevně odlišené od hodnot, které validací prošly. Tabulku s daty je možné exportovat ve formátu PDF (Portable Document Format). Ukázka tabulky s daty je na obrázku 16.

Hodnoty mohou být vizualizovány pomocí spojnicového nebo sloupcového grafu. Graf obsahuje hodnoty ze všech čidel ze zvolené skupiny, časového období a typu hodnot. Na obrázku 14 je uveden příklad sloupcového grafu, a na obrázku 15 je příklad spojnicového grafu, a na obrázku 16 je ukázka tabulky.



Obrázek 14: Ukázka sloupcového grafu

V grafu jsou barevně vyznačeny výpadky čidla a nevalidní hodnoty. Graf je možné exportovat ve formátu PNG.



Obrázek 15: Ukázka spojnicového grafu

Čidlo	Typ	Hodnota	Čas měření	Validace hodnoty	Validace čidla
1	temperature	22.4	2020-02-01 08:15:21	✔	✔
2	temperature	22.3	2020-02-01 08:15:04	✔	✔
1	temperature	24.1	2020-02-01 08:30:05	✔	✔
2	temperature	22.3	2020-02-01 08:30:05	✔	✔
1	temperature	11.3	2020-02-01 08:45:05		✔
2	temperature	22.3	2020-02-01 08:46:05	✔	✔
1	temperature	22.3	2020-02-01 08:45:05	✔	✔
2	temperature	22.3	2020-02-01 09:05:05	✔	✔
1	temperature	22.3	2020-02-01 09:00:05	✔	✔
2	temperature		2020-02-01 09:00:00	✔	✖
1	temperature	22.3	2020-02-01 09:15:05	✔	✔
1	temperature	22.3	2020-02-01 09:30:05	✔	✔

Obrázek 16: Ukázka tabulky

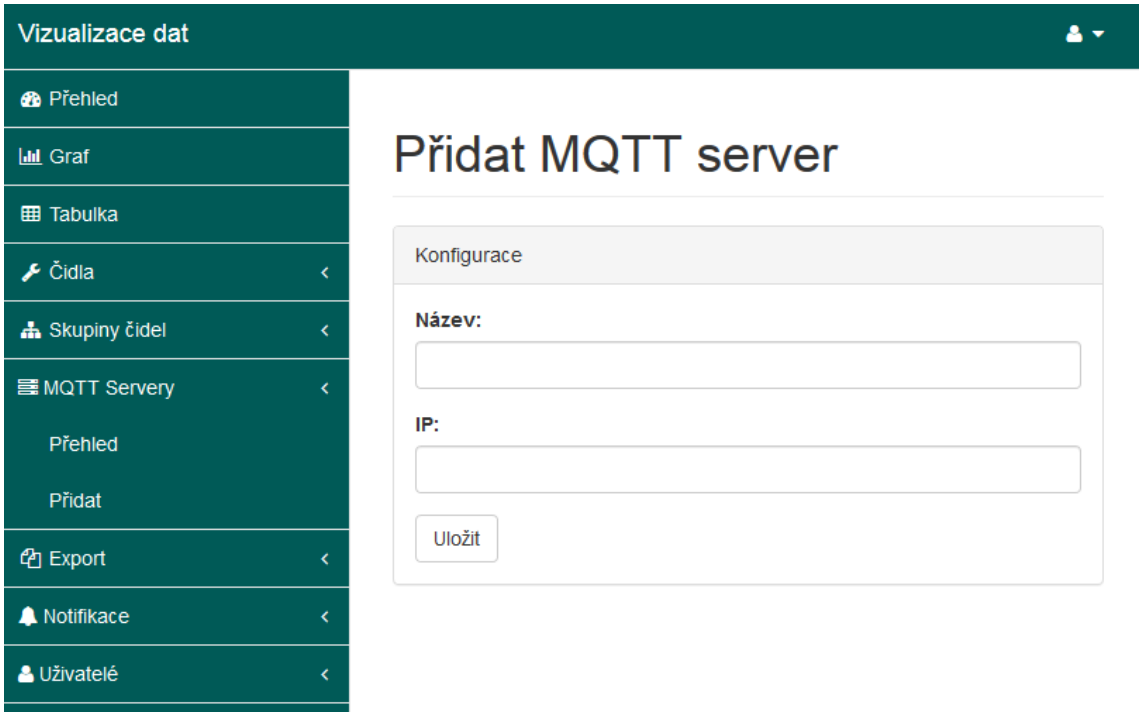
Další funkcí pro správu čidel je výpis založených čidel. Tento výpis obsahuje přehlednou tabulku se založenými čidly a akcemi, které je možné s čidly provádět. Mezi tyto akce patří smazání čidla, editace nebo export seznamu čidel ve formátu PDF.

9.1.4 MQTT broker

MQTT brokery přijímají data z čidel a přeposílají je MQTT klientovi.

Stejně jako správa čidel, tak i správa brokerů obsahuje několik základních funkcí pro jejich evidenci.

Mezi tyto funkce patří založení nového brokeru. U brokeru se evidují informace jako je název brokeru, popis a IP (Internet Protocol) adresa, na které je broker dostupný. Ukázka založení nového brokeru je na obrázku 17.



Obrázek 17: Ukázka vložení nového MQTT brokeru

Další funkcí je přehled, který obsahuje tabulku se založenými brokery a akcemi, které umožňují editaci a odebrání založeného serveru. Tento výpis je možné exportovat ve formátu PDF.

9.1.5 Notifikace

Aplikace umožňuje na základě validace hodnot a čidel odesílání upozornění na výpadek čidla nebo překročení hodnoty. Notifikace se definují pro konkrétní skupinu čidel. V systému je možné vytvářet dva typy upozornění. První typ upozornění odesílá upozornění pomocí emailu a druhý typ upozornění odesílá upozornění pomocí HTTP metody POST.

Správa notifikací obsahuje několik základních funkcí jako je vložení nové notifikace, přehled notifikací, editace notifikace a odebrání notifikace.

9.1.6 Příjem dat

Systém umožňuje příjem dat pro vizualizaci dvěma způsoby. Všechna přijatá data se ukládají do databáze, aby mohla být provedena jejich validace a vizualizace.

První způsob přijmu dat je realizován pomocí MQTT, kde jsou data z MQTT brokeru přijímána MQTT klientem. Data jsou přijímána ve formátu JSON, který obsahuje informace o čidle, ze kterého byla získána, datum a čas měření ve formátu UNIX timestamp, a seznam naměřených hodnot. Naměřené hodnoty obsahují informaci o typu hodnoty a samotnou naměřenou hodnotu.

Druhý způsob přijmu dat je pomocí HTTP metody GET. U tohoto způsobu se předává identifikátor čidla v parametru id, datum a čas měření v parametru time, typ měřené veličiny v parametru type a naměřená hodnota v parametru value.

9.1.7 Validace

Důležitou součástí systému je validace dat, která umožňuje průběžně kontrolovat naměřená data a reagovat na výpadky nebo výkyvy hodnot.

Validace je rozdělena do dvou částí. První část validace zkoumá naměřené hodnoty a odesílá upozornění na překročení nastavených limitů pro skupinu čidel. Druhá část validace zkoumá samotná čidla. Zjišťuje, zda nedošlo k jeho výpadku a nebo poruše a odesílá upozornění na výpadek nebo poruchu čidla.

- **Validace hodnoty** se provádí v několika krocích. V prvním kroku validace hodnoty se hodnota porovnává s minimální hodnotou, která je uvedena v konfiguraci skupiny čidel pro konkrétní typ veličiny. Pokud je hodnota menší než definované minimum, tak hodnota není validní a odesílá se upozornění. V druhém kroku validace se hodnota porovnává s maximální hodnotou, která je uvedena v konfiguraci skupiny čidel pro konkrétní typ veličiny. Pokud je hodnota větší než definované maximum, tak hodnota není validní a odesílá se upozornění. V posledním kroku validace se porovnává rozdíl hodnoty a předchozí naměřené hodnoty. Tento rozdíl se zjišťuje pouze v případě, že pro čidlo a typ měření existují validní data z předchozího měření stejného typu.

- **Validace čidla** se skládá ze dvou částí. V první části validace se zkoumají hodnoty od jednoho čidla a zjišťuje se, zda nedošlo k výpadku měření. V druhé části se zkoumají naměřené hodnoty z celé skupiny čidel a zjišťuje se, zda nedošlo k poruše čidla.

- **Výpadek čidla** se zjišťuje pro každé čidlo zvlášť. Čidla provádí měření a odesílání dat v pravidelných intervalech. Interval měření je definován pro všechny typy měření v konfiguraci skupiny čidel. Výpadek čidla znamená, že čidlo v očekávaném časovém intervalu nenaměřilo a neodeslalo žádnou hodnotu.

Tato část validace se provádí tak, že se postupně prochází hodnoty, které nebyly validovány, seřazené podle data a času měření vzestupně. Výpadek čidla se zjišťuje tak, že se vypočítá rozdíl mezi časem měření u validované hodnoty a časem měření u předchozí hodnoty stejného čidla a typu měření. K výpadku čidla došlo v případě, že tento rozdíl je větší než 1.5x frekvence měření.

- **Chyba čidla** se zjišťuje porovnáváním všech hodnot shodného typu a času měření u všech čidel ze skupiny čidel. U čidla došlo k chybě měření, jestliže rozdíl naměřené hodnoty a hodnoty naměřené jiným čidlem ze skupiny ve stejném čase je větší než tolerance, která je uvedena v konfiguraci skupiny čidel pro určitý typ měření.

  Porovnávání všech hodnot z měření se provádí tak, že se porovnává minimální a maximální hodnota ze všech hodnot, aby se nemusely porovnávat všechny hodnoty navzájem. Jestliže je rozdíl mezi minimální a maximální hodnotou menší než tolerance, pak jsou všechny hodnoty validní. Pokud je rozdíl větší než tolerance, pak se v případě, že existuje kromě minimální a maximální hodnoty ještě další hodnota, to znamená, že jsou hodnoty alespoň ze třech čidel, tak se zjišťuje rozdíl mezi touto hodnotou a maximální nebo minimální hodnotou a tento rozdíl se porovnává s tolerancí. Tímto způsobem lze určit zda u čidla, které naměřilo minimální nebo maximální hodnotu nedošlo k chybě.

9.2 Zdroje dat

Data ukládaná do tabulky data se získávají dvěma způsoby a to pomocí MQTT protokolu popsaného v kapitole 7.3.2 a HTTP metody GET popsané v kapitole 7.3.1. Data pro validaci a vizualizaci pomocí grafu nebo tabulky jsou načítána z databáze.

9.3 Struktura aplikace na vizualizaci dat

Aplikace je vytvořená pomocí Laravel frameworku, který má následující strukturu:

- **app** - adresář obsahuje jádro aplikace
  - **console** - adresář obsahuje vlastní příkazy
  - **exceptions** - adresář obsahuje vyjímky
  - **http** - adresář obsahuje logiku obsluhující požadavky
    - **controllers** - adresář obsahuje kontrolery
    - **middleware** - adresář obsahuje middleware třídy
  - **providers** - adresář obsahuje třídy, které se starají o nastavení zpracování požadavků nebo registraci vlastních událostí
- **bootstrap** - adresář obsahuje zavádějící soubor frameworku a mezipaměť vygenerovaných rout a balíčků

- **config** - adresář obsahuje konfiguraci projektu
- **database** - adresář obsahuje databázové migrace a třídy naplnění databázových tabulek falešnými daty
- **public** - jedná se o kořenový adresář celého webu, obsahuje soubor *index.php*, který zpracovává všechny požadavky na web
  - **css** - adresář obsahuje CSS soubory
  - **js** - adresář obahuje JavaScript soubory
- **resources**
  - **js** - adresář obsahuje nezkompilované JavaScript soubory
  - **lang** - adresář obsahuje soubory s frázemi a jejich překlady
  - **sass** - adresář obsahuje nezkompilované SASS soubory
  - **views** - adresář obsahuje pohledy
- **routes** - adresář obsahuje všechny definované routy
- **storage** - adresář obsahuje
- **tests** - adresář obsahuje všech unit a integrační testy
- **vendor** - adresář obsahuje všechny knihovny spravované přes Composer[1].

9.4 Vzhled aplikace

Vzhled aplikace je pro uživatele důležitý. Dobře navržený vzhled aplikace, který je přehledný a barevně sladěný, pak je pro uživatele ovládání aplikace intuitivní.

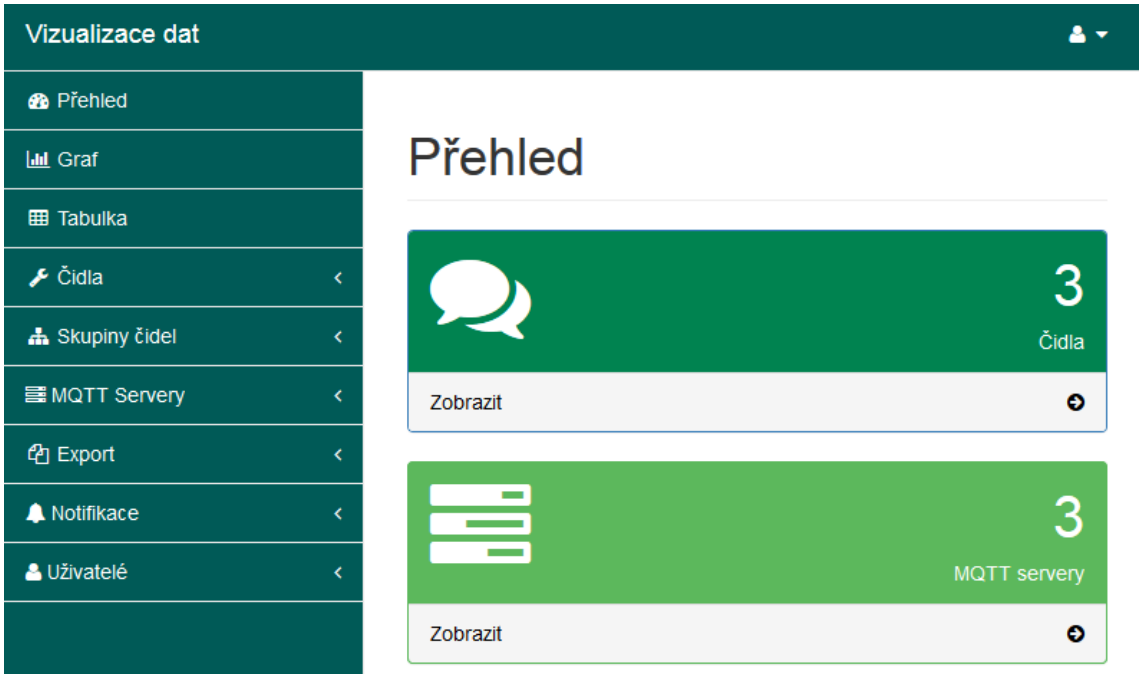
V poslední době začíná převažovat počet uživatelů, kteří přistupují k webovým aplikacím z mobilních zařízení nad těmi, kteří k přístupu používají počítač. Proto je v dnešní době důležité navrhovat a vytvářet responzivní web, aby na srovnatelné úrovni uspokojil uživatele používající mobilní zařízení s malým displejem a uživatele s velkým monitorem.

Vzhled aplikace pro vizualizaci dat je realizován pomocí open-source tématu SB Admin 2 ve verzi 2.3, které je postavené na Bootstrap 4. Ukázka vzhledu aplikace je na obrázku 18.

9.5 Rozdělení uživatelů

Systém pro vizualizaci dat obsahuje mnoho funkcí a některé z nich mohou využívat jen někteří uživatelé. Všichni uživatelé systému musí mít vytvořený uživatelský účet a musí být přihlášení, aby mohli se systémem pracovat. Uživatelé jsou rozděleni do dvou uživatelských skupin na uživatele a administrátory. Rozdělení funkcí systému a skupin uživatelů je znázorněno pomocí diagramu užití na obrázku 19.

[1] Composer je multiplatformní nástroj pro správu závislostí v PHP



Obrázek 18: Ukázka vzhledu aplikace

9.5.1 Administrátor

Administrátor nemá pro práci se systémem žádné omezení a může využívat veškeré funkce, které systém nabízí. Mezi tyto funkce patří správa uživatelů, serverů, čidel, skupin čidel, notifikací a vizualizace.

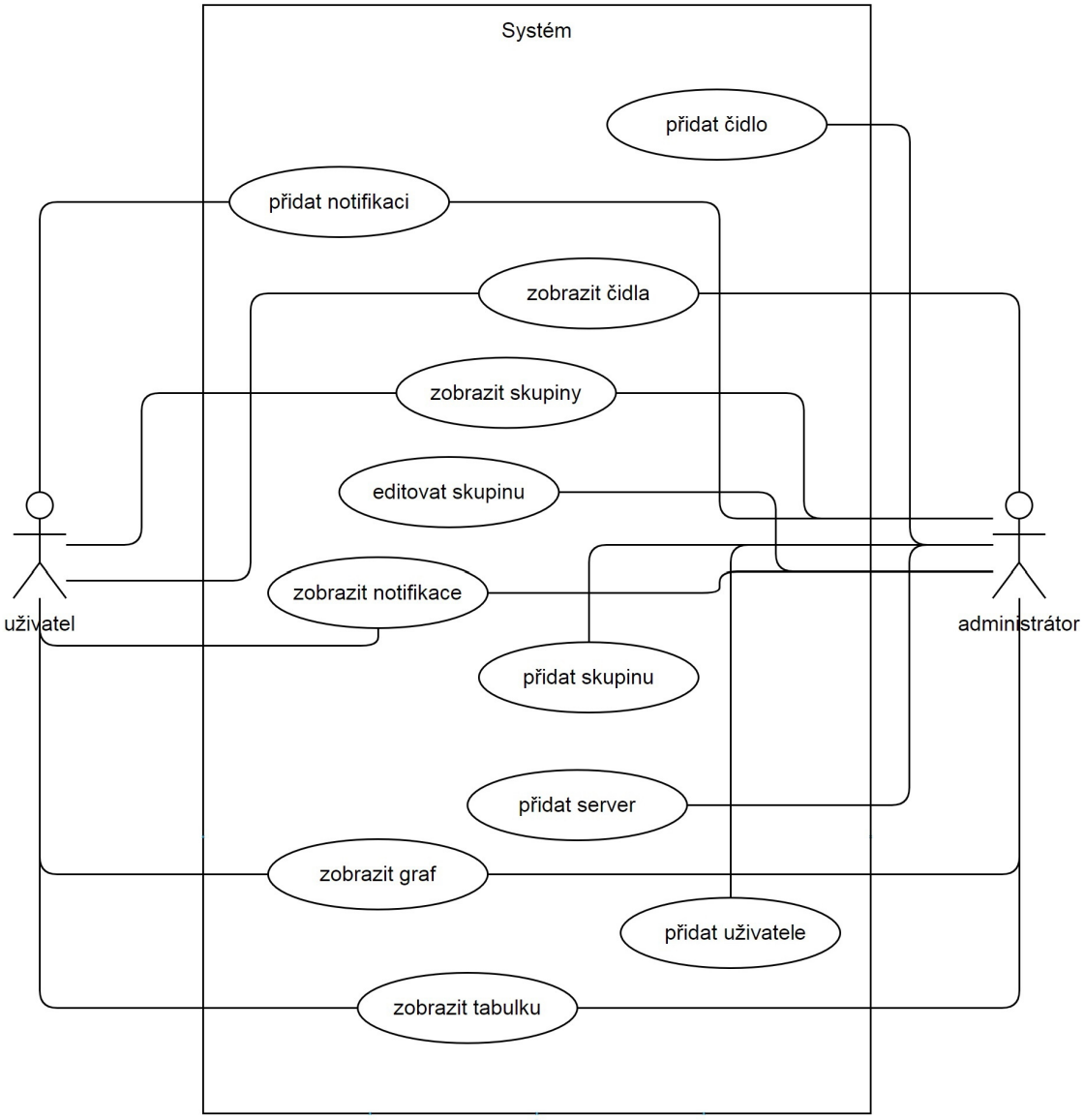
9.5.2 Uživatel

Uživatel má oprávnění využívat funkce systému pro vizualizaci dat a správu notifikací vytvořených pod svým uživatelským účtem. Uživatel si může také zobrazit seznam všech čidel, která poskytují naměřená data pro vizualizaci.

Vizualizaci dat může uživatel provádět dvěma způsoby, a to pomocí tabulky nebo grafu. Výběr dat pro vizualizaci provádí výběrem skupiny čidel a časového rozsahu. Tabulku nebo graf si může uživatel exportovat. Tabulku si může exportovat ve formátu PDF a graf ve formátu PNG.

9.6 Konfigurace nástrojů pro zajištění vysoké dostupnosti

Vysoká dostupnost aplikace pro vizualizaci dat je zajištěná pomocí HAProxy popsané v kapitole 4.4.1 a Keepalived v kapitole 4.4.2. Konfigurace vysoké dostupnosti je pomocí HAProxy a Keepalived je realizováno pouze pro zajištění vysoké dostupnosti aplikace na webových serverech. Tato konfigurace neřeší vysokou dostupnost databáze nebo proxy serveru.



Obrázek 19: Rozdělení uživatelů

Konfigurace obsahuje celkem dva webové servery a jeden proxy server. Na všech třech serverech je nainstalován operační systém Debian.

Na proxy server se HAProxy instaluje pomocí příkazu *sudo apt -y install haproxy*. Konfigurace HAProxy se provádí v souboru */etc/haproxy/haproxy.cfg*. V konfiguračním souboru se u HAProxy nastavují IP adresy obou webových serverů.

```
listen web_cluster
        balance roundrobin
        option httpclose
        option forwardfor
   option httpchk HEAD /
        cookie SERVERNAME insert indirect nocache
        server web_server_1 192.168.1.101:80 cookie web_server_1 check
        server web_server_2 192.168.1.102:80 cookie web_server_2 check
```

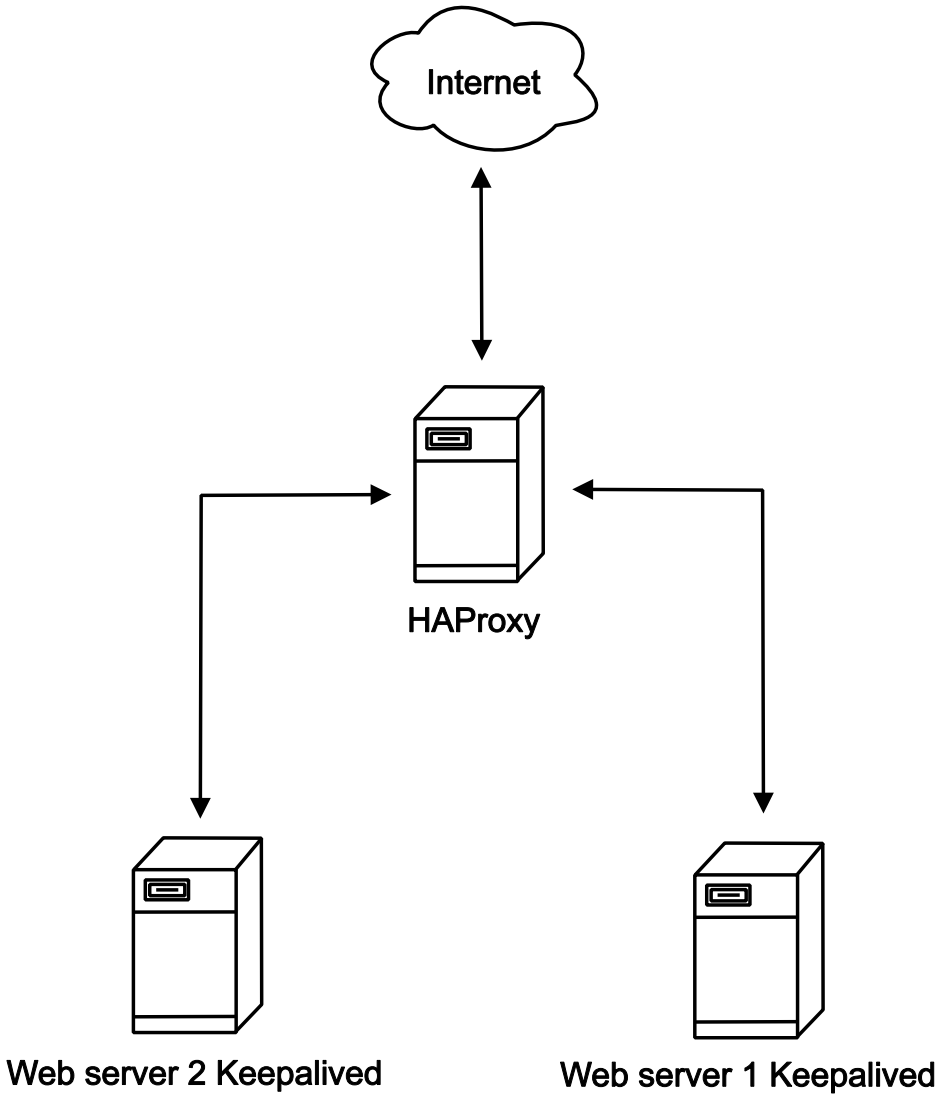
Výpis 4: Ukázka konfigurace HAProxy

Na oba webové servery se pomocí příkazu *apt-get install keepalived* nainstaluje Keepalived. Konfigurace se provádí v souboru */etc/keepalived/keepalived.conf*. Do konfigurace Keepalived se nastaví virtuální IP adresa, pod kterou bude dostupná webová aplikace pro uživatele.

```
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
   192.168.1.103
}
track_script {
   check_haproxy
}
```

Výpis 5: Ukázka konfigurace Keepalived

Na obrázku 20 je zobrazen proxy server a dva webové servery.



Obrázek 20: Kombinace HAProxy a Keepalived

10 Klient

Komunikace mezi čidly a aplikací je realizována pomocí MQTT. Zprávy předávané pomocí MQTT jsou ve formátu JSON. Předávají se celkem tři druhy zpráv. První druh zpráv odesílá aplikace s daty ke konfiguraci čidel, druhý druh zpráv odesílají čidla při potvrzení přijetí konfigurace a ve třetím druhu zpráv posílají čidla naměřená data. Vzhledem k tomu, že po přihlášení k odběru zpráv zůstává spojení aktivní, tak je část aplikace, které se stará o obsluhu MQTT řešena zvlášť mimo webovou aplikaci.

10.1 Příjem dat z čidel

Aplikace se přihlásí k odběru všech zpráv z témat, do kterých publikují čidla zprávy s naměřenými daty. Následně všechna přijatá data uloží do databáze.

10.2 Synchronizace čidel

Synchronizace se provádí tak, že aplikace publikuje zprávu ve formátu JSON, která obsahuje čas začátku měření a interval měření. Tyto zprávy se publikují do tématu, které odebírají všechny čidla ze skupiny. Během synchronizace se nejprve publikuje zpráva, kterou po přijetí všechny čidla potvrdí. V případě, že přijetí zprávy nepotvrdí všechny čidla ze skupiny, tak se zpráva s konfigurací odesílá znovu.

10.3 Struktura aplikace

Struktura aplikace je znázorněna třídním diagramem na obrázku 21. Aplikace obsahuje celkem 6 tříd, které zajišťují připojení k databázi, přenos nebo příjem dat pomocí MQTT protokolu, uložení dat do databáze a validaci formátu JSON.

10.3.1 Třída Subscriber

Třída Subscriber slouží k příhlášení k odběru zpráv z určitých témat a zpracování přijatých zpráv. Přijaté zprávy validuje pomocí třídy JsonValidator a třídy Data ukládá do databáze. Pro práci s MQTT používá MQTT klienta pro Mosquitto.

10.3.2 Třída Publisher

Třída Publisher slouží k publikování zpráv do témat. Pro práci s MQTT používá MQTT klienta pro Mosquitto.

10.3.3 Třída Synchronization

Třída Synchronization slouží k synchronizaci čidel. K předávání zpráv čidlům používá třídu Publisher. Třída obsahuje metodu getSynchronization, která vytáhne z databáze všechny aktuální

synchronizace, které ještě nebyly provedeny. Dále obsahuje metodu synchronize, která zpracovává synchronizaci a metodu updateSynchronization, která označí synchronizaci jako provedenou.

10.3.4 Třída Database

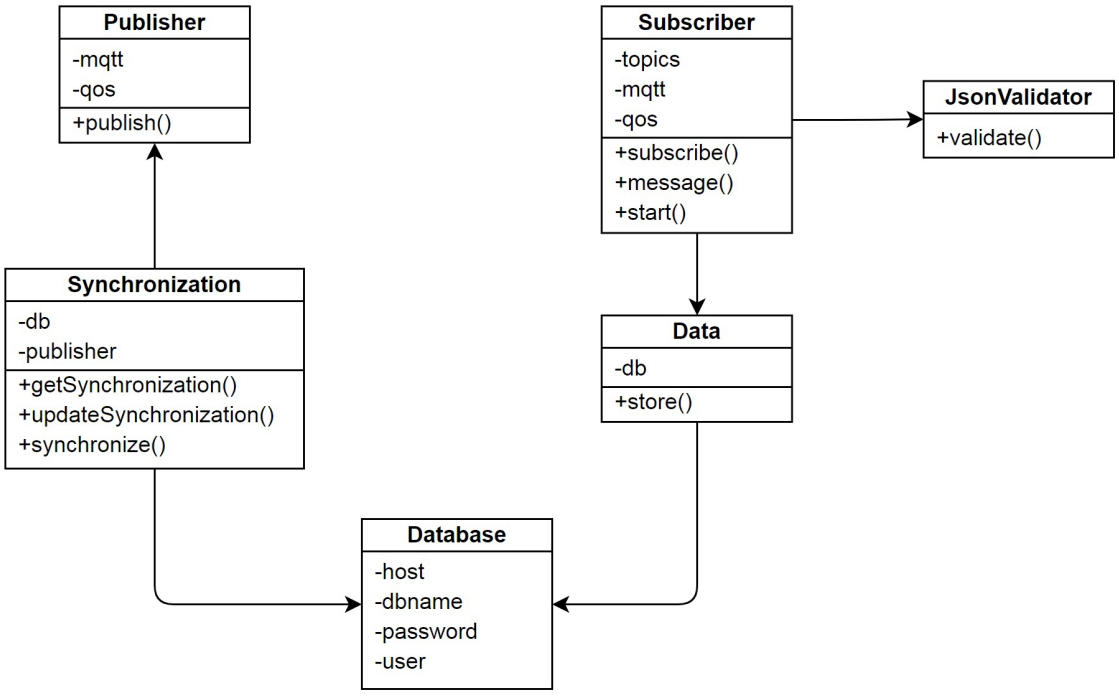
Třída obsahuje atributy, které obsahují název databáze, adresu databáze, uživatelské jméno a heslo potřebné k připojení k databázi, a vytváří spojení k databázi.

10.3.5 Třída Data

Třída data slouží k uložení všech dat z validních JSON do databázové tabulky data. Tato třída používá třídu Database a obsahuje metodu store pro uložení dat.

10.3.6 Třída JsonValidator

Třída JsonValidator slouží k základní validaci JSON. Obsahuje metodu validate, která provádí validaci struktury JSON. Struktura JSON musí obsahovat atributy sensor_id, timestamp a pole data, které obsahuje typ hodnoty a hodnotu.



Obrázek 21: Třídní diagram

11 Měření

Tato kapitola popisuje část této práce, která se zabývá měřením teploty a vlhkosti s vysokou dostupností. V kapitole je popsán návrh měření a zapojení s vysokou dostupností.

11.1 Návrh měření

Cílem bylo navrhnout aplikaci pro platformu Arduino se senzory pro měření teploty a vlhkosti s vysokou dostupností a naměřená data přenášet pomocí MQTT protokolu.

Pro realizaci byla zvolena vývojová deska Arduino UNO popsaná v kapitole 11.2. Pro měření teploty a vlhkosti byl zvolen senzor DHT22, který umožňuje měření teploty i vlhkosti. Pro přenos naměřených dat, bylo nutné připojit zařízení k internetu, a proto byl zvolen způsob připojení pomocí WiFi.

Postup měření byl navržen tak, aby bylo možné hodnoty efektivně validovat a vizualizovat. Podmínkou pro následnou validaci naměřených hodnot je určení přesného času měření. Zde se nabízí několik variant, jak určit čas, kdy byly konkrétní hodnoty naměřeny. Přesnost vnitřního oscilátoru Arduina není vhodná pro určování přesného času bez průběžné synchronizace. Pro zjištění přesného času může být použito několik modulů. Prvním modulem pro zjišťování přesného času je modul reálného času DS3231. Jedná se o extrémně přesný I2C modul s integrovaným teplotně kompenzovaným krystalovým oscilátorem a krystalem. Další možností jak zjišťovat přesný čas je použití GSM modulu nebo GPS modulu. Pro realizaci měření byl zvolen GPS modul GY-NEO6MV2, protože umožňuje nejen zjišťování přesného času, ale umožňuje určit GPS souřadnice místa, kde se měření provádělo.

11.2 Popis použitých komponent

Na následujících řádcích jsou popsány všechny komponenty, které jsou použity k realizaci měření.

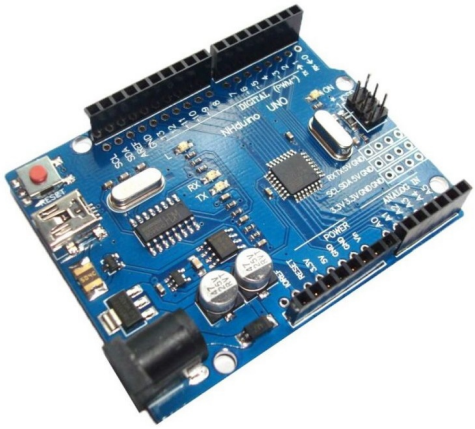
11.2.1 Digital UNO

Digital UNO je plně kompatibilní klon mikrokontrolerové vývojové desky Arduino UNO. Deska obsahuje 14 digitálních vstupních/výstupních pinů, 6 analogových vstupů, 16 MHz krystal, resetovací tlačítko, napájecí konektor, ICSP rozhraní a mikro USB konektor.

Specifikace:

- MCU: ATmega328
- USB převodník: CH340
- Vstupní napětí: 7-12V
- I/O piny: 14

- Flash 32 KB
- SRAM 2 KB
- EEPROM 1 KB
- Krystal 16 MHz



Obrázek 22: Arduino UNO

11.2.2 Digitální sensor teploty a vlhkosti DHT22

DHT22 je modul kompatibilní s platformami Arduino, Raspberry Pi a dalšími. Je umístěn v malém plastovém pouzdře o rozměrech 25x15x8 mm. Modul má celkem 4 vývody VDD, DATA, NULL a GND znázorněné na obrázku 23. Připojují se pouze tři vývody a to DATA, VDD a GND. U platforem Raspberry Pi a Arduino se na pin DATA připojuje pullup rezistor. Napájení modulu je 3,3 - 6V.
Rozsah měření:

- měření vlhkosti: 0 % $\sim$ 100 % RH
- měření teploty: -40 až +80 °C

Přesnost měření:

- měření vlhkosti: $\pm$ 2.0 % RH
- měření teploty: $\pm$ 0.5 °C

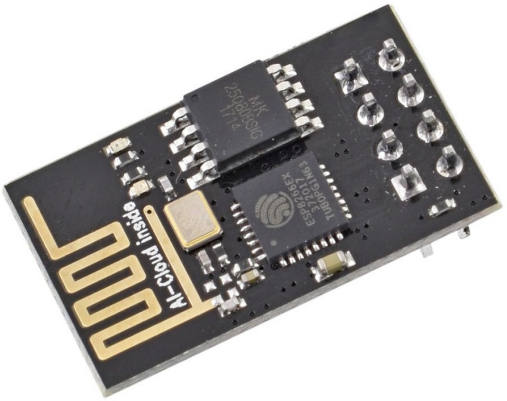
Specifikace:

- jednodrátová sběrnice WIRE
- napájení 3,3 - 6V

Obrázek 23: DHT22

11.2.3 WiFi ESP-01

WiFi modul s integrovaným mikrokontrolerem. Modul má celkem 8 pinů (GND, VCC, TDX, RDX, GP0, GP2, RST, CHPD).



Obrázek 24: Wifi modul ESP-01

Specifikace:

- napájení: 3,3V
- integrovaná anténa
- QOS management
- 32-pin QFN pouzdro
- SDIO 2.0, SPI, UART
- Integrované šifrování a zabezpečení WEP, TKIP, AES, a WAPI
- Integrovaný RF přepínač, balun, 24dBm PA, DCXO, a PMU

Síť:

- 802.11 b g n
- Wi-Fi Direct (P2P), soft-AP
- Integrovaný TCP IP stack
- Integrovaný TR switch, balun, LNA
- SDIO 2.0, SPI, UART
- Podpora 3 režimů: AP, STA, AP + STA

11.2.4 GPS Modul GY-NEO6MV2

GPS modul s anténou a vestavěnou EEPROM. Modul má celkem pět vývodů. Dva vývody pro napájení (VCC, GND), dva vývody pro data (RX, TX) a jeden vývod pro připojení antény.

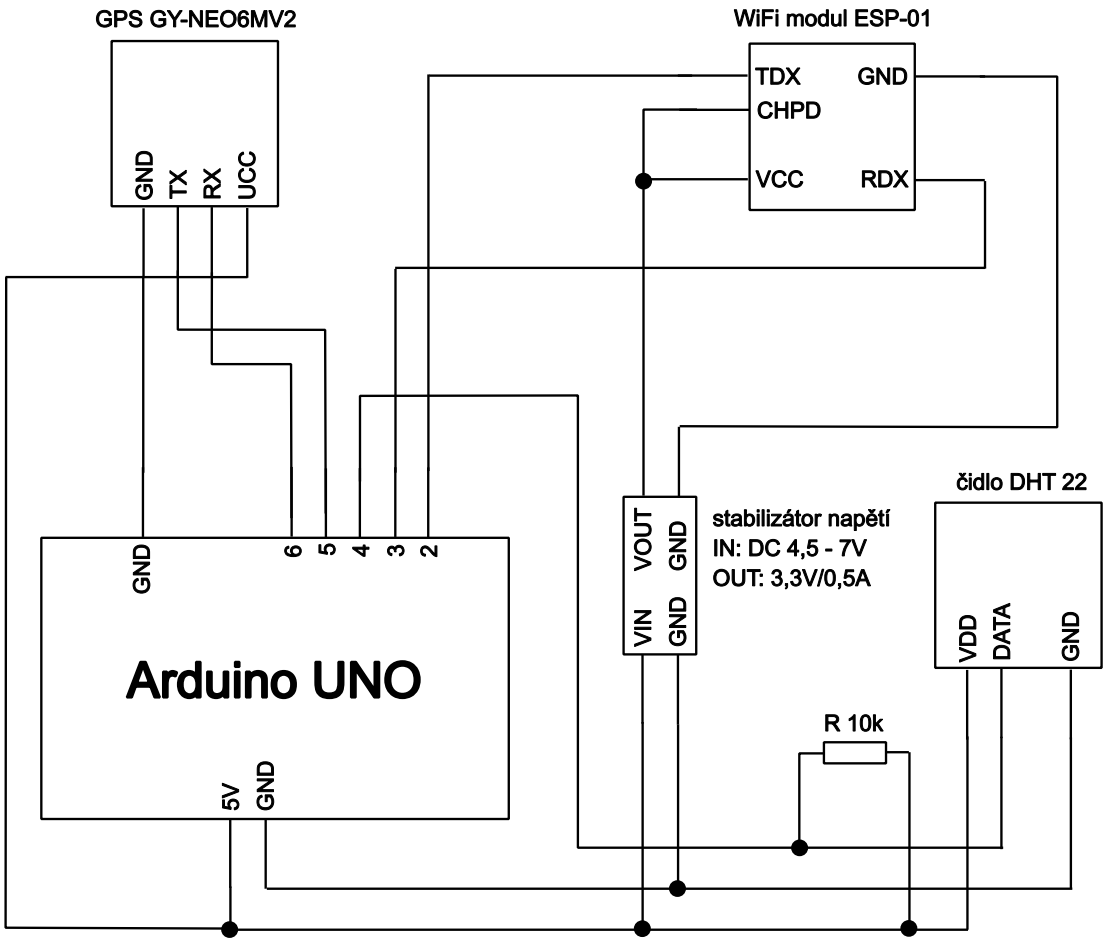
Obrázek 25: GPS GY-NEO6MV2

Specifikace:

- Napájení: 3,3V5V
- Vestavěná EEPROM paměť
- Keramická anténa
- Přenosová rychlost: 9600
- Rozhraní: RS232 TTL
- Rozměry desky: 25x35mm
- Rozměry antény: 25x25x8mm

11.3 Schéma zapojení

Na obrázku 26 je znázorněno zapojení všech modulů popsaných v kapitole 11.2.



Obrázek 26: Schéma zapojení

11.4 Přenos dat

Přenos naměřených dat je realizován pomocí formátu JSON. Jedná se o odlehčený formát pro výměnu dat, který je jednoduše čitelný a zapisovatelný pro člověka a strojově snadno analyzovatelný a generovatelný. JSON je pro výměnu dat ideální, protože je to textový formát zcela nezávislý na jazyce[15].

Příklad přenášených dat v JSON je uveden ve výpisu 6. Přenášená data obsahují id čidla, čas měření ve formátu UNIX timestamp a naměřené hodnoty. Naměřené hodnoty obsahují informace o jaký typ měření se jedná a naměřenou hodnotu. Dále je uveden příklad přenášených dat u měření teploty a vlhkosti.

```
{
   id: 1,
   time: 1585746223,
   data: [
              {'type' : 'temperature', 'value' : 20.3},
              {'type' : 'humidity', 'value' : 42.5}
          ]
}
```

Výpis 6: Příklad naměřených dat v JSON

11.5 Synchronizace času

Přesný čas je důležitý pro synchronizaci měření na čidlech a validaci naměřených hodnot. Přesnost vnitřního oscilátoru Arduina nemusí být pro dlouhodobé měření dostačující, a proto je potřeba čas v Arduinu průběžně synchronizovat.

Synchronizace času je u Arduina realizována podle přesného času z GPS (Global Positioning System) modulu. Čas v Arduinu se podle času z GPS modulu synchronizuje na začátku programu a pak průběžně každých šedesát minut.

11.6 Synchronizace měření

Začátek měření začíná v určitý čas a měření se provádí v pravidelném časovém intervalu. Začátek a interval měření je pro celou skupinu čidel stejný.

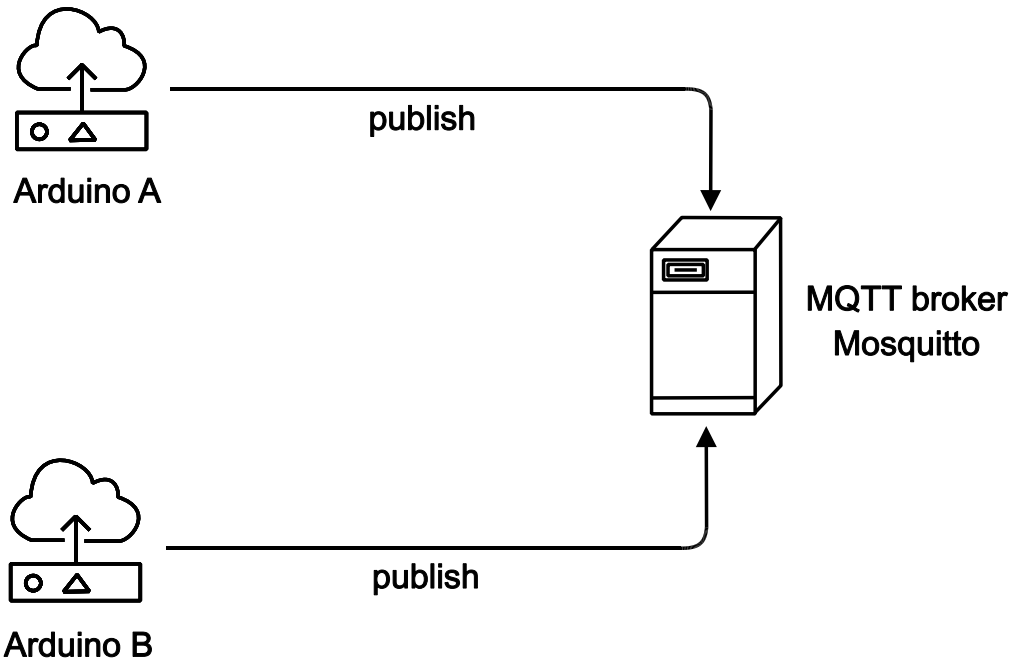
Čidlo se při zapnutí příhlásí MQTT brokeru k odběru zpráv, které obsahují ve fromátu JSON čas začátku měření ve formátu UNIX timestamp a interval měření v minutách. Jakmile čidlo přijme zprávu, tak nastaví interval měření a podle přijatého času začátku měření začne měřit. Následuje výpis 7 s příkladem zprávy s konfigurací měření.

```
{
   start: 1585746223,
   frequency: 15
}
```

Výpis 7: Příklad konfigurace

11.7 Vysoká dostupnost měření

Vysoká dostupnost měření je realizována pomocí paralelního měření více čidel zobrazených na obrázku 27. Začátek měření je u všech čidel provádějících stejné měření nastaven pomocí synchronizace popsané v kapitole 11.6.



Obrázek 27: Vysoká dostupnost měření

12 Závěr

Cílem této práce bylo provést rešerši v oblasti vysoké dostupnosti a vizualizace dat, navrhnout a vytvořit systém pro vizualizaci dat s vysokou dostupností.

V teoretické části práce je rešerše v oblasti vysoké dostupnosti, která zahrnuje spolehlivost a bezporuchovost zapojení součástek, definice a klasifikace poruchy, popis a měření vysoké dostupnosti.

Navržený systém je implementován v PHP pomocí Laravel frameworku a data jsou uložena v MySQL databázi. V systému jsou uživatelé rozdělení do dvou skupin na administrátory a uživatele. Systém umožňuje vizualizovat data dvěma způsoby a to pomocí grafu nebo tabulky. Součástí nástroje na vizualizaci je validace, které se skládá ze dvou částí a to z části, která validuje naměřené hodnoty a umožňuje odesílání upozornění v případě, že je naměřená hodnota mimo nastavený rozsah a validace čidla, které zajišťuje zjišťování výpadku nebo poruchy čidla a umožňuje odesílání upozornění v případě, že u čidla došlo k poruše nebo výpadku. Výstupy z obou validací jsou použity při vizualizaci dat pomocí tabulky a grafu tak, že jsou hodnoty, které jsou mimo stanovený rozsah jsou v grafu i tabulce podbarveny červenou barvou a výpadky čidla jsou v tabulce a grafu podbarveny barvou šedou. Vygenerovanou tabulku je možné exportovat ve formátu PDF a graf ve formátu PNG. Jedním z požadavků na nástroj pro vizualizaci dat byla jeho vysoká dostupnost. Jelikož se jedná o webovou aplikaci, tak byla pro zajištění vysoké dostupnosti zvolena HaProxy a Keepalived. Webová aplikace je pak nasazená na hlavním a záložním serveru s operačním systémem Debian a v případě, že dojde k výpadku hlavního serveru, tak dojde k převzetí požadavků záložním serverem aniž by uživatel zaznamenal výpadek dostupnosti webové aplikace.

Součástí práce bylo také navrhnout a realizovat aplikaci pro platformu Arduino se senzory pro měření teploty a vlhkosti. V rámci návrhu aplikace bylo potřeba vybrat moduly pro realizaci aplikace a navrhnout formát, ve kterém se budou data posílat. Navržená aplikace je realizována pomocí platformy Arduino UNO a několika modulů jako je GPS nebo WiFi, které umožňují připojení klonu k internetu a zajištění přesného času měření. Naměřené hodnoty se odesílají pomocí protokolu MQTT. Zajištění vysoké dostupnosti měření je realizováno pomocí paralelního měření dvou čidel meřících stejnou veličinu, a je možné použít i více než dvě čidla.

Navržený systém pro vizualizaci dat s vysokou dostupností společně s aplikací pro platformu Arduino by našel praktické využití například v zoologické zahradě. Skupina alespoň dvou čidel by byla instalována v místnosti, která slouží jako líheň a je velice důležité, aby v ní byla teplota a případně i vlhkost udržována na určité hodnotě. Ošetřovatelé, kteří by se starali o líheň by si v systému pro vizualizaci nastavili upozornění pro skupinu čidel v líhni. V případě, že by například v zimě přestalo fungovat topení a teplota by klesla pod určitou hodnotu, tak by byl ošetřovatel včas informován.

Literatura

[1] *HOLUB*, Rudolf a Zdeněk VINTR. Spolehlivost letadlové techniky: elektronická učebnice [online]. Brno, 2001. Dostupné z WWW: <http://lu.fme.vutbr.cz/files/SpolehlivostLetadloveTechniky.pdf>.

[2] *Tisková zpráva – TK 5. 6. (robot da Vinci v MOÚ)* [online]. Dostupné z: <https://www.mou.cz/tz-robot-da-vinci-v-mou-5-6-2018/f2314>.

[3] *LIONS*, Prof. J. L. ARIANE 5: Flight 501 Failure [online]. 19. 7. 1996. Dostupné z WWW: <http://sunnyday.mit.edu/nasa-class/Ariane5-report.html>.

[4] *SOMMERVILLE*, Ian. Softwarové inženýrství. Brno: Computer Press, 2013. ISBN 978-80-251-3826-7

[5] *JANUROVÁ*, Kateřina. TEORIE PRAVDĚPODOBNOSTI: 2. cvičení [online]. Dostupné z WWW: <https://homel.vsb.cz/~jan939/STA/cviceni/02_Pravdepodobnost.pdf>.

[6] *Chart.js*. Dostupné z WWW: <https://www.chartjs.org/>.

[7] *MQTT* Dostupné z WWW: <http://mqtt.org/>.

[8] *Internet věcí a protokol MQTT*. IoT portál [online]. 14. 2. 2017. Dostupné z WWW: <https://www.iot-portal.cz/2017/02/14/internet-veci-a-protokol-mqtt/>.

[9] *MQTT*. IoT portál [online]. 24. 5. 2016. Dostupné z WWW: <https://www.iot-portal.cz/2016/05/24/mqtt/>.

[10] *MALÝ*, Martin. Protokol MQTT: komunikační standard pro IoT. ROOT.CZ [online]. 14. 2. 2017. Dostupné z WWW: <https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/>.

[11] *BRIŠ*, Radim a Martina LITSCHMANNOVÁ. STATISTIKA II.: učební text [online]. 2007. Dostupné z WWW: <http://www.elearn.vsb.cz/archivcd/FEI/STA2/Statistika%202.pdf>.

[12] *What is Keepalived* [online]. Dostupné z WWW: <https://www.keepalived.org/>.

[13] *HAProxy*: The Reliable, High Performance TCP/HTTP Load Balancer [online]. Dostupné z WWW: <http://www.haproxy.org/>.

[14] *SB Admin 2* Dostupné z WWW: <https://startbootstrap.com/themes/sb-admin-2/>.

[15] *Introducing JSON* [online]. Dostupné z WWW: <https://www.json.org/json-en.html>.

[16] *PUŽMANOVÁ*, Rita. TCP/IP v kostce. 2., upr. a rozš. vyd. České Budějovice: Kopp, 2009. ISBN 978-80-7232-388-3.

[17] *ORACLE*. High Availability Concepts and Best Practices [online]. Dostupné z WWW: <https://docs.oracle.com/cd/A91202_01/901_doc/rac.901/a89867/pshavdtl.htm>.

[18] *TORELL*, Wendy a Victor AVELAR. Mean Time Between Failure: Explanation and Standards: White Paper 78 [online]. Dostupné z WWW: <https://www.controldesign.com/assets/11WPpdf/110516-Schneider-mean-time-between-failure.pdf>.

[19] *Laravel*. Dostupné z WWW: <https://laravel.com/docs/6.x>.

[20] *Grafana*. Dostupné z WWW: <https://grafana.com/>.

[21] *Kibana*. Dostupné z WWW: <https://www.elastic.co/kibana>.

[22] *Power BI*. Dostupné z WWW: <https://powerbi.microsoft.com/>.